ADAPTABLE LONG-TERM OPTIMIZATION OF DRY CASK STORAGE LOADING PATTERNS

A Dissertation

by

KRISTINA YANCEY SPENCER

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Pavel V. Tsvetkov
Committee Members,	Kenneth L. Peddicord
	Sean M. McDeavitt
	Sergiy Butenko
Head of Department,	Yassin A. Hassan

December 2017

Major Subject: Nuclear Engineering

Copyright 2017 Kristina Yancey Spencer

ABSTRACT

To address the evolving needs of dry storage, this research developed an optimization methodology to identify loading configurations to minimize the number of casks, their heat load, and the time when they meet transportation requirements. The motivation was to investigate strategies that balance and reduce risk over the lifetime of a site's reactor(s).

The dry cask loading problem was formulated as an adaptable dynamic bin packing problem, accommodating different site and cask limits in broadly-defined constraints. A new method was developed to address its complexities, named the GRASP-enabled adaptive multiobjective memetic algorithm with partial clustering (GAMMA-PC). This method embeds greedy randomized adaptive search procedures in a multiobjective evolutionary algorithm with local search techniques and partial decomposition of the objective space during crossover.

GAMMA-PC was demonstrated through integration with the unified database from the Used Fuel Systems group at Oak Ridge National Laboratory to optimize simulated loading configurations for Vermont Yankee, Comanche Peak, and Zion Nuclear Power Stations. Its performance was evaluated through comparisons to test solutions and to the real Zion loading configuration. GAMMA-PC produced diverse solutions that dominated the testing sets. The improvement was concentrated in the average heat load, and the third objective function was shown to be sensitive to individual assembly characteristics.

The results suggested the usefulness of GAMMA-PC for utilities considering long-term goals. They showed that more diverse cask loadings and strategic placements of empty positions can be used to reduce initial heat loads. Moving to a higher

capacity cask increases loading flexibility but can result in transportation delays. Long-term planning enables a more thorough consideration of the trade-offs involved in any decision.

This research contributes one of the first in-depth studies of the dry cask loading problem. It expands the current treatment of assembly selection over longer time-frames and meets user-defined requirements. It is also one of the first tri-objective dynamic bin packing problems, and the first to pack items with time-dependent characteristics. Future work should focus on refining the objectives and incorporating uncertainty. With its adaptable structure, GAMMA-PC is a promising new metaheuristic for this task and for dynamic bin packing problems in general.

ACKNOWLEDGEMENTS

First, I would like to thank my committee chair and advisor Dr. Pavel V. Tsvetkov. When I started as his undergraduate research fellow, I did not imagine that I would spend the next nine years in his mentorship. I sincerely appreciate all the encouragement and advice, from connecting me with my Fulbright affiliation in Switzerland to cheering me on as I wondered if I would ever be done debugging my code over the past year. He is one of the most dedicated advisors that I know of, and I am glad to be his student.

I would like to thank my advisor from Oak Ridge National Laboratory and special committee member Dr. Joshua J. Jarrell. He took a chance on my research idea, and I am so grateful for his advice and encouragement over the past year. I would also like to thank the other researchers I met at ORNL, especially Dr. Kaushik Banerjee who helped me immensely in integrating GAMMA-PC with UNF-ST&DARDS.

I would like to thank my other committee members for their support and teaching over the years. I would know nothing about optimization if I had not taken Dr. Butenko's class, I always enjoyed the thought-provoking discussions in Dr. McDeavitt's classes, and I appreciated whenever Dr. Peddicord checked in on me during my year in Switzerland. I would also like to thank all of the professors in the department for believing in me and teaching me over the past eleven years. I want to extend that gratitude to my fellow students, to the Tsvetkov research group, and to the student chapters, especially TAMU Women in Nuclear. The nuclear engineering department feels like home to me, and I am happy to have shared it with you over the past decade.

I would like to thank my family for their love, help, and support over the years.

I am forever indebted to my parents for giving me the opportunities and experiences that made me who I am. I am grateful to my siblings for their friendship and to my family-in-law for their warm acceptance. I am also grateful to my nieces and nephews for reminding me to look at the world like it is new.

Finally, I would like to express my deepest gratitude to my husband Jonathan Spencer. I appreciate his silliness, selflessness, and understanding, especially about the need for cuddles from our two fluffy cats. He has been my biggest cheerleader over the past several years, and I could not have finished this doctorate without his support.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professors Pavel V. Tsvetkov (advisor), Kenneth L. Peddicord, and Sean M. McDeavitt of the Department of Nuclear Engineering, Professor Sergiy Butenko of the Department of Industrial Engineering, and Dr. Joshua J. Jarrell of Oak Ridge National Laboratory (external co-advisor). Dr. Jarrell currently works at Idaho National Laboratory.

The unified database UNF-ST&DARDS is maintained by the Used Fuel Systems group at Oak Ridge National Laboratory for the Department of Energy's Office of Nuclear Energy. The queries to import data from it were developed in part by Dr. Kaushik Banerjee and Paul Miller of Oak Ridge National Laboratory.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

This research was supported by the National Science Foundation under Grant No. 1252521 and by the US Department of Energy, Office of Nuclear Energy, under contract number DE-AC05-00OR22725 (Jarrell), Texas A&M University (Tsvetkov), and by the ORNL GO! Fellowship Program (Spencer). The dissertation was completed with the support of Philanthropic Education Organization Janes Hines Scholar award.

NOMENCLATURE

ALARA	as low as reasonably achievable
BWR	Boiling Water Reactor
C	Celsius
CoC	Certificate of Compliance
DBPP	dynamic bin packing problem
GAMMA-PC	GRASP-enabled adaptive multiobjective memetic algorithm with partial clustering
GRASP	greedy randomized adaptive seach procedure
GWd	gigawatt-days
IAEA	International Atomic Energy Agency
ISFSI	Independent Spent Fuel Storage Installation
kW	kilowatt
LLC	Limited Liability Corporation
MOEA	multiobjective evolutionary algorithm
MOEA/D	multiobjective evolutionary algorithm with decomposition
MOEPSO	multiobjective evolutionary particle swarm optimization
MOMA	multiobjective memetic algorithm
MOMAD	memetic algorithm based on decomposition
MPa	megapascal, the SI unit of pressure
MPC	multipurpose canister
MTU	metric tonne of uranium
NRC	Nuclear Regulatory Commission

NSGA-II	nondominated sorting genetic algorithm II
OC	oldest, coldest
PWR	Pressurized Water Reactor
RCL	restricted candidate list
SI	International System of Units
SQL	structured query language
Sv	sievert, the SI unit of ionizing radiation
U.S.	United States
UNF-ST&DARDS	Used Nuclear Fuel - Storage, Transportation & Disposal Analysis Resource and Data System

Variables & Mathematical Notation:

$\mathbf{x}$	Decision vector
i	Index used to correspond to bin number
j	Index used to correspond to item number
x	Packing matrix
y	Array denoting the bins in use
t_{fill}	Array denoting the time at which each bin is packed
$\mathbf{F}(\mathbf{x})$	Objective vector
Ω	Feasible region
$u \succ v$	Objective vector u dominates vector v
N	Number of items to be packed
$\overline{M}$	Theoretical maximum number of bins to pack items
β	Cardinality restriction
$BU_{s,max}$	Maximum allowable assembly average burnup for a storage cask

$BU_{t,max}$	Maximum allowable assembly average burnup for a transport cask
C_{cask}	Capacity of the cask used to pack the assemblies
C_{pool}	Capacity of the spent fuel pool
F_p	Maximum number of casks allowed to be filled within one period
$H_{s,max}$	Maximum allowable total decay heat for a storage cask
$H_{t,max}$	Maximum allowable total decay heat for a transport cask
m	Number of objective functions contained in the objective vector
P	Set of solutions to be sent to genetic operators
P_{EA}	External archive of nondominated solutions
Q	Set of solutions produced by genetic operators and found through local search
$t_{trans,min}$	Array containing the time at which each cask meets the transportation guidelines

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Nuclear reactors produce large amounts of clean, reliable power and leave behind relatively little waste, which remains highly radioactive and thermally hot for many years. When the majority of the current reactor fleet was built, it was envisioned that either reprocessing or a long-term repository would be operational before spent fuel pools reached capacity, so pools were not designed to hold used fuel generated over the entire reactor lifetime [104]. However, these assumptions proved to be incorrect. Used fuel has remained on-site and in increasingly crowded pools, making additional storage capacity necessary.

The majority of nuclear power plants in the United States (U.S.) have already established an Independent Spent Fuel Storage Installation (ISFSI). Six more are currently pursuing a general license for one [105], and only three have not declared their intention to use dry storage, each of which only operates a single reactor.[1] Regardless of future policy decisions about U.S. nuclear waste management, dry storage will continue to be used for decades. As evidence, the Nuclear Regulatory Commission (NRC) considered the environmental impact of on-site dry storage for an indefinite amount of time beyond a reactor's licensed life in making the Continued Storage Rule [106].

While utilities are focused on their immediate storage needs, all on-site fuel will eventually need to be safely transported to a new location(s) for future disposition. To plan for both the possibility of long-term storage as well as transportation, caretakers of used fuel would benefit from considering the competing and cooperating aspects of these pathways. The International Atomic Energy Agency (IAEA) identi-

[1] These three plants are listed in NRC document ML16286A019: Three Mile Island, Wolf Creek, and Shearon Harris.

fied a number of areas in which dry storage could be optimized with respect to long term storage, such as selecting better cask materials, reducing uncertainties and conservatisms, and increasing the burnup/age acceptability of assemblies [107]. One area in particular would impact both dry storage and transportation: the selection of assemblies to be moved into dry storage. The IAEA recommended that utilities move away from a coldest/oldest-first loading strategy and toward a more diverse mixture of assemblies. The research discussed in this dissertation explores this recommendation and develops a methodology to determine loading patterns that would be optimal for metrics relevant to worker safety and transportation, accounting for current and future inventory.

1.1 Dry Storage

The U.S. does not have a standardized dry cask design. Instead, a few companies offer an array of dry storage systems from which utilities can chose. Major differences between models include cask material, orientation, and purpose [108]. The NRC regulates dry storage designs in the U.S. and issues a Certificate of Compliance (CoC) for each specific model. The CoC describes the uses, surveillance requirements, and administrative controls needed to safely utilize the dry cask system.

The NRC has granted CoCs for eighteen dry storage systems, fourteen of which are still active [109]:

1. the VSC-24 system (CoC 1007),
2. the FuelSolutions spent fuel management system (CoC 1026),
3. the NUHOMS horizontal modular system (CoC 1004),
4. the Advanced NUHOMS horizontal modular storage system (CoC 1029),
5. the NUHOMS HD horizontal modular storage system (CoC 1030),

6. the HI-STAR 100 cask system (CoC 1008),
7. the HI-STORM 100 cask system (CoC 1014),
8. the HI-STORM FW system (CoC 1032),
9. the HI-STORM UMAX canister storage system (CoC 1040),
10. the TN-32 dry storage cask (CoC 1021),
11. the TN-68 dry storage cask (CoC 1027),
12. the NAC-UMS universal storage system (CoC 1015),
13. the NAC-MPC system (CoC 1025),
14. the MAGNASTOR system (CoC 1031).

Excluding the variations of the NUHOMS design, all of the approved cask systems are stored vertically [110]. The standard cask size holds 24 pressurized water reactor (PWR) or 68 boiling water reactor (BWR) assemblies, but in the past decade, companies have moved toward higher capacity casks holding 32 assemblies or more [111]. The highest capacity designs, the HI-STORM FW and the MAGNASTOR systems, can accomodate 37 PWR assemblies or 89 and 87 BWR assemblies, respectively [112].

Companies have also moved toward more flexibility in their models. Earlier models, such as the VSC-24, were only designed for storage, while newer models have been designed for both storage and transportation, such as the NAC-MPC system [110]. Most of the dual-purpose designs incorporate more flexibility into the design by separating an inner container, or canister, that holds the fuel together from the outer boundary layer. This allows the canister to be placed into either a storage overpack or a transport cask, both of which have been certified by the NRC for their intended purpose. Alternatively, the HI-STAR 100 system stores the fuel "directly in the transportation package" [110].

The CoCs establish the requirements for acceptable assemblies in each cask system. Before 2002, the NRC only allowed enrichment limits to be set using the conservative assumption of fresh fuel during the criticality safety analysis. This is not a problem for lower-capacity casks that include flux traps or increased fixed-poison concentrations, but higher-capacity casks made room for more fuel at the expense of neutron absorbers [113]. For storage, this is not a limitation. The NRC does not consider the possibility of fresh-water ingress into sealed storage casks to be credible, so the storage cask would be in its most critical state during loading procedures [114]. Therefore, applicants are allowed to take credit for the boron in the spent fuel pool water to calculate the criticality of the cask. However, the NRC does consider the fresh water ingress of a transport cask to be a significant risk, so the fresh fuel assumption disqualified most used fuel from being transported in high-capacity casks.

CoC requirements for assembly selection have evolved with improvements in the technical understanding of actinide and fission product behavior in used fuel [115]. The agency updated the transportability of assemblies by allowing for the use of burnup credit in PWR safety analyses, which takes into account the reactivity reduction from fuel depletion [116]. The NRC approved the use of actinide-only burnup credit in 2002 and extended their recommendations to include some fission products in 2012 [111]. This change increased the percentage of acceptable assemblies for loading into transportation casks from less than 20% to about 90% [117].

1.2 Dry Cask Planning Efforts

With the evolution of dry cask systems, the need for the selection of "a sensible mixture" of used fuel assemblies is even more important [107]. The guidelines to accept assemblies are more nuanced than they were in the 20th century. While CoCs

for storage casks have not incorporated burnup credit yet, CoCs for transportation casks have started to replace enrichment limits with loading curves of the minimum burnup as a function of initial enrichment and cooling time [111]. Most dry casks have only been loaded with storage in mind, which might make transporting used fuel more complicated in the future. Even if burnup credit were applied to those that have already been loaded, only 90% of the assemblies would meet the loading criteria [117]. Given the large number of PWR assemblies already in storage, a substantial number might have to be moved into lower-capacity or more heavily engineered casks. BWR assemblies could face the same predicament since they have more complex power histories and less validated data to support the use of burnup credit. While ISFSI license holders can apply for an exemption, and regulations and analyses will continue to evolve, employing a dry cask loading strategy now that mitigates the need for either will help alleviate financial risk. With about 3/4 of used fuel sitting in pools [118] and the inventory possibly doubling before the end of U.S. reactors' licensed lives [119], many more assemblies will need to be moved into dry storage. It is important to move them wisely.

To aid in the intelligent transfer of used fuel into dry storage, this research develops an algorithm to optimize the loading of the casks. Little work has been published in this area. Table 1.1 lists the four programs that are currently available and shows that existing dry cask loading programs are limited. All of the programs only select used fuel from the current pool inventory. None of the papers published about Studsvik's MARLA [120] or about the Electric Power Research Institute's CASKLOADER [122] disclose the method used to perform their optimization. While the other two do describe their method, neither uses a state-of-the-art optimization methodology that would enable efficient computation in such a large search space. Moreover, they do not include regulatory requirements as constraints, so their meth-

Table 1.1: Previous Dry Cask Loading Pattern Optimization Studies.

Reference	Objective	Decision Space	Method
Studsvik's MARLA [120]	Clear space in spent fuel pool & meet CoC requirements.	Loading pattern of dry casks for a number of given dates – available fuel only.	Unstated.
EPRI's CASKLOADER [121], [122], [123]	Minimize cost based on schedule constraints.	Loading pattern of individual dry casks – available fuel only.	Unstated.
Jožef Stefan Institute [61]	Minimize number of canisters and time in interim storage.	Loading of dry canisters in deep geological repository.	Heuristic sorting algorithm with "remove and reinsert" local search.
KAERI [124]	Decrease bore hole spacing.	Loading of dry canisters in deep geological repository.	Regression analysis and manual evaluation.

ods are not useful for real world application. MARLA and CASKLOADER are the only choices in Table 1.1 that help utilities pick assemblies that meet the CoC requirements, but they focus on short-term goals, and it is uncertain how flexible their algorithms can be.

1.3 Research Motivation

This research is motivated to develop a methodology that is more robust and adaptable than the current dry cask loading programs. It takes a more comprehensive and long-term view than those listed in Table 1.1 as it considers used fuel generated over the entire core lifetime. It is also the first to optimize based on long-term transportation needs. The algorithm finds optimal loading patterns that balance the competing goals of dry storage: higher capacities and lower dose rates for both workers and the environment. Finally, the dry cask loading problem is treated more formally here than in previous research, both in describing the problem

mathematically and in developing a state-of-the-art optimization method to solve the problem.

This work should benefit the nuclear industry and society. The optimization of assembly selection should improve the efficiency of operations at a nuclear power plant. Even if a plant is doing better than the "coldest/oldest-first loading" mentioned by the IAEA [107], the best tools right now only plan for the next several loadings and are limited to the assemblies currently stored in the spent fuel pool. Finding better loading patterns should decrease the cumulative dose rate due to transfer procedures over the reactor lifetime and should remove a few more of the hotter assemblies from the pool on a regular basis, reducing thermal stresses in wet storage.

Optimization now will also make it easier to transport the casks to a central storage facility in the future. The decommissioning of Zion Nuclear Power Station can be used as an example. ZionSolutions, LLC, began restoring the site in 2010 [123]. The company developed its own semi-manual optimization strategy to load 2226 fuel assemblies into 61 MAGNASTOR casks. They used CASKLOADER to ensure that each cask met CoC requirements, but it was only able to load a single basket at a time. To achieve their goal of limiting off-site dose rates, they established heuristic rules about the location of source terms on the dry cask pad and synchronized CASKLOADER with additional calculations to define the loading configuration. If casks are loaded according to an optimal pattern during operation, much less effort should be needed at the end of a reactor's life.

Finally, the optimization of dry cask loading patterns can have a positive impact on a future interim storage facility or repository with regard to the distribution of heat. By including some hotter assemblies in casks loaded during the reactor lifetime, the backlog of the hottest assemblies at the beginning of decommissioning would

be mitigated to an extent controlled by the structure of the objective vectors. The research presented here is more focused on finding better solutions for transportation, but constraints and objectives could be modified in the future to represent repository-specific needs. In any case, a better distribution of heat would help lower peak temperatures, allowing for a tighter spacing of waste packages [125].

1.4 The Used Nuclear Fuel Data System

The optimization tool produced in this research was integrated into the Used Nuclear Fuel-Storage, Transportation & Disposal Analysis Resource and Data System (UNF-ST&DARDS) [126]. UNF-ST&DARDS is a comprehensive data and analysis tool that provides the most accurate information about U.S. used fuel available. Fig. 1.1 shows its organizational structure. The data portion of the tool is contained in a unified database and can be split into five main categories:

1. fuel assembly discharge information,
2. fuel assembly design data,
3. reactor-specific operation data,
4. cask design and loading data,
5. infrastructure and logistics-related data.

This information is organized using relational structured query language (SQL) data tables within a MySQL database and is maintained by the Used Fuel Systems group at Oak Ridge National Laboratory for the Department of Energy's Office of Nuclear Energy [1].

UNF-ST&DARDS is the foundation of the accuracy and specificity of the optimization discussed in this dissertation. It includes data for over 150,000 used fuel assemblies as a function of decay and several hundred currently loaded cask

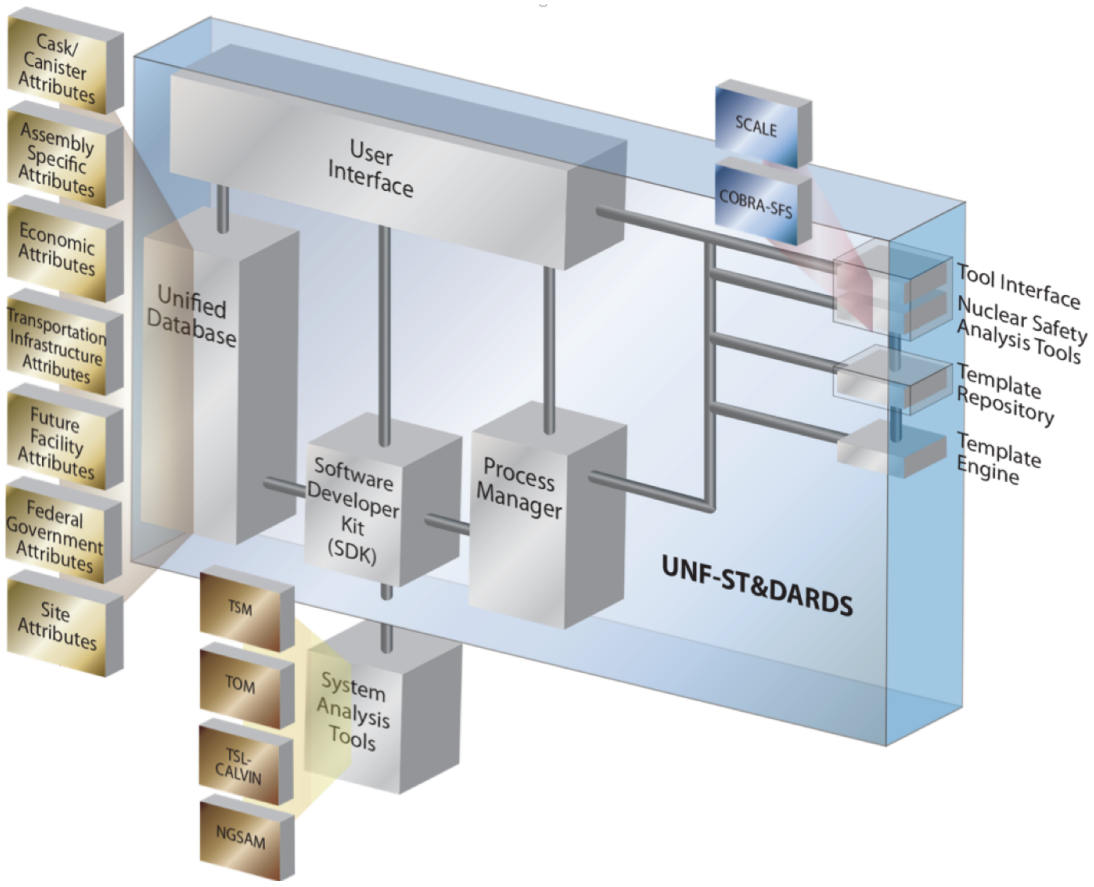


Figure 1.1: Structure of UNF-ST&DARDS. Reprinted courtesy of Oak Ridge National Laboratory, US Dept. of Energy [1].

systems [126]. The optimization problem presented here cannot be solved using representative assemblies because a few representative models cannot give a clear picture of the significant variation found in historical fuel assemblies [127]. Therefore, detailed assembly-specific data is a prerequisite to find optimal loading patterns. UNF-ST&DARDS also includes the ability to integrate with other codes and can be interfaced via the command line, so the optimization algorithm can pull the necessary information from the database to populate loading patterns for an ISFSI.

1.5 The Dry Cask Loading Problem

The dry cask loading problem is a hard combinatorial multiobjective optimization problem. Any algorithm used to solve it must handle the task of sorting thousands of individual assemblies into a set of casks based on a combination of simple and complex constraints. This is also a dynamic problem since the casks would not all be loaded simultaneously. It is an uncertain problem, both from uncertain fuel assembly characteristics and from uncertain future projections. The search space is large, and multiple solutions exist. These features have been difficult for traditional optimization methods to handle and emphasize the need to use a state-of-the-art optimization algorithm to solve the problem.

The problem can be visualized as shown in Fig. 1.2. The algorithm selects assemblies from a dictionary of used fuel assemblies specific to a given pool. The dictionary includes the fuel currently in the pool and the projected discharges that will enter it throughout the lifetime of the reactor. The algorithm finds feasible combinations of assemblies in casks and promotes solutions that minimize the objectives. Since the entire core lifetime is considered, the algorithm also determines the time at which transfer procedures are performed for each cask.

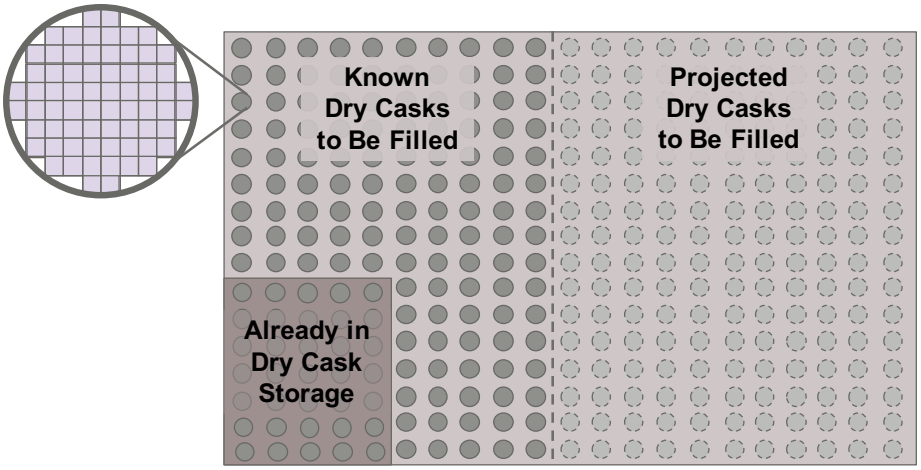


Figure 1.2: Illustration of dry cask loading pattern defining the problem search space.

1.6 Dissertation Objectives

The primary goal of this research was to develop a novel method to determine optimal loading patterns of used fuel assemblies in dry cask storage, accounting for current and future inventory and optimizing over metrics relevant to worker safety and transportation of used fuel. The method was implemented in a computational tool for demonstration and analysis.

This research can be broken into four main tasks:

1. Formulation of the mathematical structure and parameterization of the dry cask loading problem.

2. Development of a new optimization methodology to solve the multiobjective dry cask loading problem.

3. Implementation of the optimization methodology into a computational tool for integration with UNF-ST&DARDS.

4. Demonstration and analysis of the methodology in dry cask loading test cases.

The computational tool was implemented using Python, which is useful for an object-oriented problem and has an established interface to work with MySQL. The tool was developed to ensure that its code would be adaptable, maintainable, extensible, and modular. These features will allow future users to optimize different objectives, add new constraints or heuristic rules, and maintain the tool in connection with updates to UNF-ST&DARDS. It also separates the optimization algorithm from the problem being solved so that as needs evolve, the updates can be managed independently of improvements in the underlying algorithmic framework.

1.7 Dissertation Organization

Section 2 describes the approach used in this work. It provides background on combinatorial problems, gives an overview of algorithms used to solve them, and defines fundamental optimization terms that will be used throughout the dissertation. The mathematical structure for the dry cask loading problem is then developed in detail, formulating the decision variables, the objectives, and the constraints used to bound the search space.

Section 3 presents the new methodology developed in this research. It gives an outline of the algorithmic framework and presents detailed descriptions of novel steps included in the method. It also discusses the development of specific modules in the optimization tool used to demonstrate the new method.

Section 4 demonstrates the performance of the new method and presents an analysis of test cases used to evaluate the effectiveness of the tool.

Finally, Sec. 5 summarizes new and significant contributions by this work and suggests areas for improvement in the future.

2. METHODOLOGY

The dry cask loading problem is a mixed-variable multiobjective, combinatorial optimization problem. It features the competing goals of minimizing the number of casks needed, reducing initial heat loads, and minimizing the storage time required before transport. Section 2 discusses previous approaches to similar problems, defines necessary terms, and develops the mathematical formulation of the dry cask loading problem.

2.1 Optimization of Combinatorial Problems

With three objectives, the dry cask loading problem follows the general format of the 3-dimensional minimization problem given in Eq. 2.1.

$$\begin{aligned} minimize \quad & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})), \\ s.t. \quad & \mathbf{x} \in \mathbf{\Omega}. \end{aligned} \tag{2.1}$$

Here, $f_\theta(\mathbf{x})$ denotes objective θ, which is a function of the decision variable $\mathbf{x}$. The decision variable $\mathbf{x}$ belongs to the **feasible region** $\mathbf{\Omega}$, which bounds the search space to values that satisfy a set of constraints. The function $\mathbf{F}$ translates $\mathbf{x}$ into the **objective space**, and the value of a single objective f_θ is called a **fitness value**.

Multiobjective problems are characterized by competing objectives, so a single solution cannot minimize (or maximize) all of the objective functions simultaneously. Consequently, the best solutions involve trade-offs between the different objectives. This set of solutions is called the **Pareto set**, and the image of the Pareto set in the objective space is called the **Pareto front**. A feasible solution is proven to be part of the Pareto set by showing that it is not *dominated* by any other feasible solution,

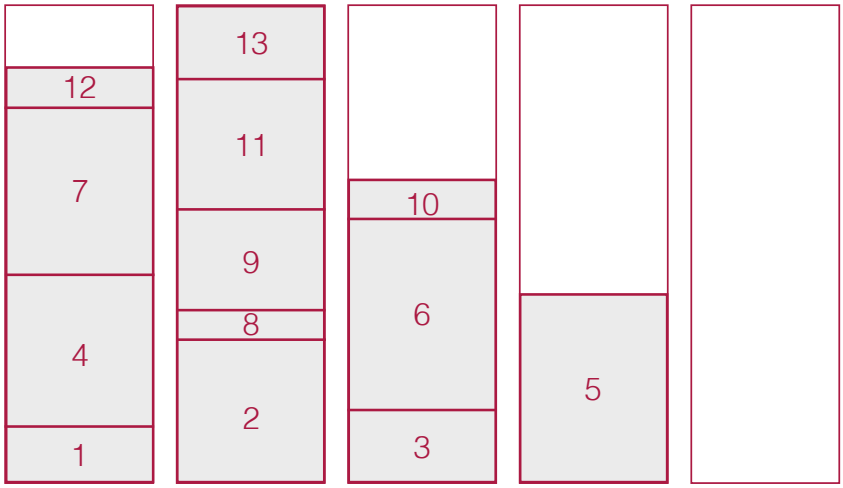


Figure 2.1: Illustration of bin packing.

defined formally in Def. 2.1 [128].

Definition 2.1. *In a multiobjective problem with m objectives, an objective vector $u = (u_1, ..., u_m)^T$* ***dominates*** *another vector $v = (v_1, ..., v_m)^T$ iff $\forall \theta \in \{1, ..., m\}$, $u_\theta \leq v_\theta$ and $u \neq v$, written $u \succ v$.*

An optimization algorithm may not reach the true Pareto front during its finite computation. Consquently, the computed set of nondominated feasible solutions is called an **approximation set** [129].

The dry cask loading problem falls within the general category of bin packing problems. These are combinatorial problems that pack a number of items N into as few bins as possible within constraints such as weight or size limits. Fig. 2.1 presents an illustration of this type of problem. To represent this mathematically, the decision variables for a bin packing problem use binary forms, shown in Eqs. 2.2 and 2.3.

$$x_{ij} \in \{0, 1\}, \ \forall \ i = \{1, ..., \overline{M}\}, j = \{1, ..., N\} \tag{2.2}$$

$$y_i \in \{0,1\},\ \forall\, i \in \{1,...,\overline{M}\} \tag{2.3}$$

Here, if item j is assigned to bin i, then $x_{ij} = 1$. Otherwise, the variable is set to 0. The matrix x is complemented by the array y that indicates which bins are in use ($y_i = 1$) and which are closed ($y_i = 0$). The constant $\overline{M}$ represents the theoretical maximum number of bins and is usually set at $\overline{M} = N$ in the literature [103, 130].

The first mathematical formulation of a multiobjective bin packing problem was published in 2007 with the objectives of minimizing both the number of bins and the average deviation from the center of gravity in the bins [130]. Since then, many bi-objective bin packing problems have been studied. Given the flexible nature of bin packing problems, they have represented subjects as concrete as packing boxes for transport and as abstract as cloud computing [131].

2.1.1 Optimization Algorithms

Traditionally, bin packing problems have been solved using approximate algorithms, such as the *First-Fit* and *Best-Fit* methods [132]. Exact algorithms exist for very small problems but become too costly as the problem size increases from medium to large packing problems. Approximate algorithms, on the other hand, produce near-optimal solutions in $O(n \log n)$ time, significantly reducing the computational burden.

The loading study performed by Žerovnik et. al. used this type of approach to pack spent fuel assemblies from the Krško nuclear power plant into canisters for deep geological disposition. The inventory data for the optimization was generated using the CORD-2 package for burnup, mass, and initial material composition and the ORIGEN 2.1 code for thermal power as a function of cooling time [61]. The assemblies were then packed into canisters using variations of a largest-fit heuristic constrained by a maximum thermal limit. The study compared the heuristic re-

sults to a theoretical minimum number of canisters and showed that the heuristic achieved solutions within 5% of the theoretical minimum. However, for multiobjective problems with more complex constraints, approximate algorithms work best when combined with a method suited for the multidimensional objective space.

Since the available literature on dry cask loading optimization is limited, other nuclear-related combinatorial optimization studies were reviewed. The majority of this research focused on core reload and shuffling schemes [12, 15, 21, 23, 25, 33, 39, 44, 47, 49, 66, 69, 74, 77, 78, 80, 82, 83, 86–88, 94], although other areas such as fuel cycle optimization [76, 84] and maintenance planning [99] were explored. Many of the studies did not elaborate on why one algorithm was chosen over another, so statistics about their choices are used here to describe how the nuclear field has approached combinatorial optimization problems.

To get a better understanding of the trends, 100 optimization papers in the nuclear field were sampled [2–101]. Fig. 2.2 shows a bar chart separating the sample into different types of optimization methodologies. Evolutionary algorithms are the most popular choice, followed by simulated annealing and particle swarm-based approaches. Each of the categories in the graph have strengths and weaknesses. For example, evolutionary algorithms are robust and have good **exploration** abilities, meaning that their search for feasible solutions avoids getting trapped in local minima (or maxima). However, they tend to have poor **exploitation** abilities, meaning that the search has trouble finding the precise global minimum (or maximum). On the other hand, particle swarm optimization approaches tend to have powerful exploitative abilities but poor exploration abilities [78]. Simulated annealing can vary between these extremes based on a set of sensitive parameter settings [66].

It is possible to combine the advantages of different algorithms by utilizing **ensemble learning** in which decision-making strategies or learning algorithms work

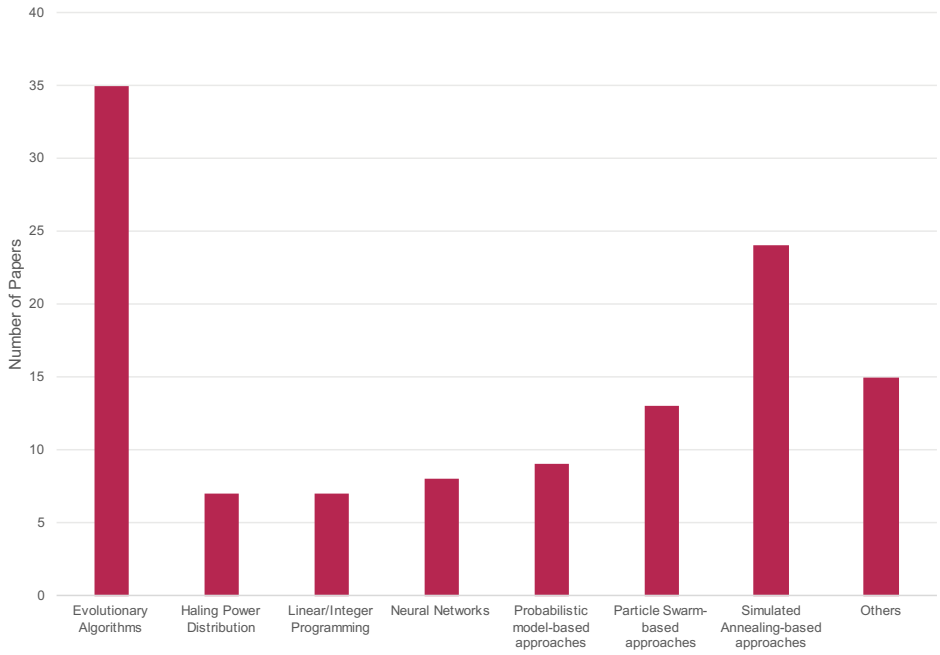


Figure 2.2: Distribution of optimization methodologies used for combinatorial problems in nuclear engineering. This bar chart was created based on a survey of 100 papers [2–101].

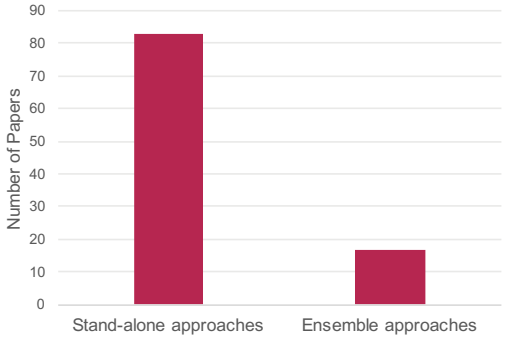


Figure 2.3: Comparison of stand-alone versus ensemble optimization approaches in nuclear engineering. This bar chart was created based on a survey of 100 papers [2–101].

together to improve performance [133]. Fig. 2.3 shows that most of the surveyed sample only used individual methods, and less than 20% used ensemble approaches. Almost all of the studies using ensemble approaches have been published in the past five years, and it is likely that this trend will continue to grow.

2.1.2 Ensemble Approaches to Multiobjective Packing Problems

Beyond the nuclear field, multiobjective evolutionary algorithms (MOEAs) have been the most popular method for solving multiobjective optimization problems [128]. This is largely due to their population-based nature, allowing them to quickly generate an approximation to the Pareto front. However, as an extension of evolutionary algorithms, MOEAs suffer from a weak exploitation ability and can be slow to converge. Using an ensemble approach can greatly improve this limitation.

For multiobjective bin backing problems, a common ensemble is to combine the heuristics of approximate algorithms with MOEAs, allowing the heuristic to perform the packing and the MOEA to perform the search. Given the nature of each type of method, the two use different forms of the decision variable, requiring a translation whenever the mode of operation is changed.

Three decision variable formats are useful here:

- the matrix representation,
- the chromosome representation,
- and the variable length bin representation.

The matrix representation was defined in Eqs. 2.2 and 2.3 and is used whenever a fitness value is calculated. The **chromosome** representation is a real-valued permu-

tation of item indices. An example is given in Eq. 2.4.

$$genes = [3, 8, \cdots, 64, 23, \cdots, N] \tag{2.4}$$

Each number *genes* is associated with a specific item j and therefore with column j in the x-matrix. The MOEA uses this representation to search for new solutions. The **variable length representation** is a real-valued list of sublists containing item indices. An example is given in Eq. 2.5.

$$v.l.rep. = \begin{matrix} [1, 2, 3, 7, 8], \\ [9, 14, 56], \\ \vdots \\ [86, 123, 185, N] \end{matrix} \tag{2.5}$$

The variable length representation contains as many lists as there are open bins, indicated by $y_i = 1$. Each sublist in the list corresponds to one bin and contains the indices of the items located in that bin, indicated by $x_{ij} = 1$. Depending on how the approximate algorithm is designed, it can use either the matrix representation or the variable length representation to pack the items into bins.

One example of this type of ensemble is the combination of the nondominated sorting genetic algorithm II (NSGA-II) with the least loaded heuristic presented in [103]. NSGA-II is one of the most successful MOEAs [128]. It is particularly known for its *fast-non-dominated-sort* procedure and its *crowding-distance-assignment*, reproduced in Tables 2.1 and 2.2, respectively [102]. To adapt the algorithm for a bin packing problem, NSGA-II is combined with the least loaded heuristic, reproduced in Table 2.3 [103]. This heuristic takes the chromosome representations produced by the genetic operations in NSGA-II and performs the packing with a strategy to

Table 2.1: The *fast-non-dominated-sort* algorithm described in [102]. This procedure sorts solutions into fronts, where $\mathcal{F}_1$ represents the approximate set, and each subsequent front contains nondominated solutions in relation to higher-number fronts.

```
1   for each p ∈ P:
2       S_p = ∅
3       n_p = 0                          domination counter
4       for each q ∈ P:
5           if (p ≺ q):
6               S_p = S_p ∪ {q}
7           else if (q ≺ p):
8               n_p = n_p + 1
9       if n_p = 0:                      p belongs to F_1
10          p_rank = 1
11          F_1 = F_1 ∪ {p}
12  i = 1                                Initialize the front counter
13  while F_i ≠ ∅:
14      Q = ∅
15      for each p ∈ F_i:
16          for each q ∈ S_p:
17              n_q = n_q − 1
18              if n_q = 0:              q belongs to F_{i+1}
19                  q_rank = i + 1
20                  Q = Q ∪ {q}
21      i = i + 1
22      F_i = Q
```

Table 2.2: The *crowding-distance-assignment* algorithm described in [102]. This operator calculates the distance among the solutions in the objective space. The crowding distance is used as a secondary parameter during binary selection to prioritize solutions with larger distance values and thereby improve diversity.

```
1   l = |I|                                  number of solutions in I
2   for each k, set I[i]_distance = 0
3   for each objective m:
4       I = sort(I, m)                       Sort I by objective m
5       I[1]_distance = I[l]_distance = ∞
6       for k = 2 to (l − 1):
7           I[k]_distance = I[k]_distance + (I[k + 1].m − I[k − 1].m) /
                                             (f_m^max − f_m^min)
```

Table 2.3: The least loaded heuristic described in [103].

```
     Input: A permutation of items σ = (1, ..., N), lower bound LB
     Output: A packing solution
1    m = LB
2    for j = 1 to N:
3         for i = 1 to m:
4              Try to pack item j into the least loaded bin
5              if Packable:
6                   Add item j to the bin i
7                   break
8         if j = m + 1:
9              m = m + 1
10             Add item j to the new bin m
11        Update the order of the bins to keep the list sorted.
12   end
```

balance the load between the bins.

Recent work has focused on combining MOEAs with local search methods, the combination of which is called a **memetic** algorithm. This type of ensemble offers better convergence and better accuracy than the evolutionary approach alone [128]. Most of the work in this area has focused on continuous problems, but memetic algorithms have been applied to combinatorial problems as well. One study combined NSGA-II with hill-climbing local search techniques to solve a multiobjective knapsack problem, a cousin of the bin packing problem [134]. Their findings showed drastic improvements in performance with the addition of local search techniques but only when the local search was built upon problem-specific knowledge.

2.2 Defining the Dry Cask Loading Optimization Paradigm

Using the basic format of a bin packing problem, the dry cask loading problem can be formulated. This section describes the decision variables used in this work, formulates the objective vector, and discusses the constraints used to ensure that the loading algorithm produces solutions that meet user-defined requirements.

2.2.1 Decision Variables

This work expands on the standard bin packing problem decision vector format to include time as a decision variable. Equation 2.6 presents the general form of the decision vector $\mathbf{x}$ used in this research, comprised of three distinct decision variables.

$$\mathbf{x} = \begin{bmatrix} x & y & t_{fill} \end{bmatrix} \tag{2.6}$$

These variables are expanded in Eq. 2.7. Variables x and y are the same matrix form as was established in Sec. 2.1. Variable t_{fill} is a new, continuous variable indicating the time each cask is loaded with used fuel assemblies.

$$x = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,j} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,j} & \cdots & x_{2,N} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ x_{i,1} & x_{i,2} & \cdots & x_{i,j} & \cdots & x_{i,N} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ x_{\overline{M},1} & x_{\overline{M},2} & \cdots & x_{\overline{M},j} & \cdots & x_{\overline{M},N} \end{bmatrix}, \; y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_{\overline{M}} \end{bmatrix}, \; t_{fill} = \begin{bmatrix} t_{fill,1} \\ t_{fill,2} \\ \vdots \\ t_{fill,i} \\ \vdots \\ t_{fill,\overline{M}} \end{bmatrix} \tag{2.7}$$

The index i is used to identify the dry cask number, and j is used to identify the assembly number. The index ranges are given in Eq. 2.8.

$$i \in \{1, ..., \overline{M}\}, j \in \{1, ..., N\} \tag{2.8}$$

Here, $\overline{M}$ represents the theoretical maximum number of casks, and N represents the number of used fuel assemblies to be sorted. While many bin packing problems set the theoretical maximum at N, this research uses a smaller value. Loading used nuclear fuel into dry cask storage is a complex, intensive process, so it is beneficial

to avoid solutions that include mostly empty casks. The ideal scenario for plant operations would be to fill every cask to capacity, although there may be situations that call for leaving empty spaces in a canister. Therefore, $\overline{M}$ is set to allow for each cask to be a quarter empty if the assemblies were spread evenly throughout the canisters, given by Eq. 2.9.

$$\overline{M} = \left\lceil \frac{N}{0.75 C_{cask}} \right\rceil \tag{2.9}$$

The variable C_{cask} represents the capacity of one canister. When more than one cask type is used during the optimization, the smallest capacity is used for C_{cask} in Eq. 2.9.

2.2.2 Objective Vector

In taking a long-term view of used fuel storage systems, three objective metrics were used to improve worker safety and the future transportability of dry casks:

1. minimize the number of casks needed to store the fuel,
2. minimize the average initial heat load in each cask, and
3. minimize the length of time for casks to meet transportation dose limits.

These three objectives reflect economic, safety, and social concerns. The combined objective vector is formulated mathematically in Eq. 2.10.

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} \sum_{i=1}^{\overline{M}} y_i \\ \frac{1}{c} \sum_{i=1}^{c} w_{r,i} Q_i \\ max(t_{trans,min,i}, i = \{1, ..., c\}) \end{pmatrix} \tag{2.10}$$

Here, $w_{r,i}$ is a weighting factor for the initial heat in cask i, Q_i (kW). The variable $t_{trans,min,i}$ represents the earliest date that cask i would meet transportation dose

limits, and c is the number of casks in use. Its value is equal to the fitness value of the first objective, as shown in Eq. 2.11.

$$c = f_1(\mathbf{x}) = \sum_{i=1}^{\overline{M}} y_i \tag{2.11}$$

This is the standard objective function for all bin packing problems.

Beyond the structure of a standard bin packing problem, the first objective function has economic implications for the utility. Each cask has costs associated with it, from its purchase to its maintenance and surveillance. While the number of casks needed is not necessarily linearly related to the total cost of dry storage [135], it does serve as a simple metric to gauge the resources needed for each solution found by the algorithm. The number of casks in an ISFSI also helps determine how many shipments are needed to move the fuel off-site.

2.2.2.1 Minimizing Initial Cask Heat Loads

The second objective function seeks to minimize the average initial heat load of the casks and is reiterated in Eq. 2.12.

$$f_2(\mathbf{x}) = \frac{1}{c} \sum_{i=1}^{c} w_{r,i} Q_i \tag{2.12}$$

The initial heat in each cask Q_i is a linear sum of the decay heat values H for each used fuel assembly in the cask at time $t_{fill,i}$, as shown in Eq. 2.13.

$$Q_i = \sum_{j=1}^{N} x_{ij} \cdot a_j . H(t_{fill,i}) \tag{2.13}$$

The decay heat H is retrieved from variable a_j, which is called from a dictionary containing assembly-specific information imported from UNF-ST&DARDS, such as

initial enrichment values (wt.%) and burnup levels (MWd/MTU). To improve the computational efficiency of the algorithm, the decay heat is found by linear interpolation of a table of projected values from UNF-ST&DARDS for a_j.

The weighting factor $w_{r,i}$ in Eq. 2.12 reduces the impact of casks loaded in the future compared to those loaded in the near term when evaluating this objective. It does this through the economic idea of a social discount rate r_f and is formulated in Eq. 2.14.

$$w_{r,i} = \left(\frac{1}{1+r_f}\right)^{(t_{fill,i}-t_0)} \tag{2.14}$$

The exponential in Eq. 2.14 is the difference in years between the time cask i is filled and the beginning of the timeline t_0. The optimization timeline is initialized at the discharge date of the last currently existing batch of assemblies added to the spent fuel pool unless otherwise instructed.

The use of a discount weighting factor in a bin packing problem is unique to this research. It is employed for two main reasons. The first is that the cask loadings toward the end of the reactor lifetime are less certain than those in the near future, so the discount rate puts more emphasis on near-term loadings. The second is that it is assumed that technology and operations will continue to improve as they have over the past few decades.

Setting the rate of discount can be sensitive. An r_f value of 2.0% per year has been used in economic analyses of nuclear waste management, which is more conservative than in public infrastructure analyses due to the need to ensure resources for proper waste management [136, 137]. The Swedish Radiation Safety Authority has even suggested rates as low as 0.5% to 1.0% to mirror the risk management of the Swedish bond market [137]. Considering the generally conservative nature of nuclear utilities, an r_f value of 0.5% was used in this research.

While the main goal of this objective is to reduce the average heat load and thereby improve thermal performance of the casks, it also serves as a convenient proxy for the dose that workers receive during transfer procedures. Both the decay heat and the radioactivity of the fuel are functions of nuclide decay, and their magnitudes tend to decrease together [138]. While it is not a perfect proxy, the improvement in computational efficiency for such a large multiobjective optimization problem makes decay heat a practical subsitute for dose. Therefore, the second objective helps align solutions with the industry practice of keeping its radiological impact as low as reasonably achievable (ALARA) during transfer procedures, which is especially important since it represents the back-end process where nuclear plant workers receive the single highest dose [121].

2.2.2.2 Reducing the Storage Time Before Transport

The third objective function works to minimize the length of time until the casks meet transportation CoC limits. This is represented in Eq. 2.15.

$$f_3(\mathbf{x}) = max(t_{trans,min,i}, i = \{1, ..., c\}) \tag{2.15}$$

The vector $t_{rail,min}$ contains the times at which each open cask meets the transportation limits, and the maximum value in the array determines the value for the third objective. This is similar to the way many multiobjective bin packing problems have a secondary objective to minimize the tallest bin height, although calculating $t_{rail,min}$ is more complicated.

Three different transportation CoC constraints are used to find the required storage time for each cask: dose, total decay heat, and per assembly decay heat limits. The maximum time required to meet all the limits is the earliest time that a cask

would be ready for transport, as shown in Eq. 2.16.

$$t_{trans,min,i} = max(t_{LC,i}, t_{HT,i}, t_{HA,i}) \tag{2.16}$$

The time values compared in Eq. 2.16 represent the time that cask i meets the transportation loading curve, the total decay heat limit, and the per assembly decay heat limits, respectively.

The time to meet the transportation loading curve is found first with Eq. 2.17.

$$t_{LC,i} = \begin{cases} max(a_j.t_{cool}), & \textbf{if } a_j.(enr, BU) \in LC_{cask}, \forall\ j \in vlrep[i] \\ t_{dose,i} & otherwise \end{cases} \tag{2.17}$$

Here, t_{cool} is the cooling time required by the transporation loading curve LC_{cask} for the unique enrichment (enr) - burnup (BU) combination of assembly j. The (enr, BU) tuple must fall above the enrichment and below the burnup curves within LC_{cask} to be acceptable [139]. The LC_{cask} curve was developed to ensure that the cask meets applicable federal regulations for shielding during transport, such as a dose rate limit of 0.1 mSv/hr at 2 meters from the cask [140]. Therefore, the variable $t_{LC,i}$ is set by the first line in Eq. 2.17 if all the assemblies in bin i fall within this region. Otherwise, $t_{LC,i}$ is set by solving Eq. 2.18.

$$\dot{D}_{target} - \frac{\sum_{j=1}^{N} x_{ij} \cdot a_j.\dot{D}(t_{dose,i})}{\sum_{j=1}^{N} x_{ij}} = 0 \tag{2.18}$$

In Eq. 2.18, the time $t_{dose,i}$ is found when the average dose rate of the assemblies in the bin is equal to the target dose rate $\dot{D}_{target}$, found through comparison to the LC_{cask}. The dose rate for each individual assembly is calculated through linear

interpolation of a table of projected dose rates imported from UNF-ST&DARDS. The modified regula-falsi method was used to find $t_{dose,i}$ [141].

The time to meet the total decay heat limit is found using Eq. 2.19.

$$H_{t,max} - \sum_{j=1}^{N} x_{ij} \cdot a_j.H(t_{HT,i}) = 0 \tag{2.19}$$

Here, $H_{t,max}$ is the total decay heat limit for the cask during transportation. Initially, the left hand side of Eq. 2.19 is evaluated using $t_{LC,i}$, and if the result is positive, finding the true $t_{HT,i}$ is avoided since the limit is respected. However, if the result is negative, the root of Eq. 2.19 must be found, again using the modified regula-falsi method.

Finally, the time to meet the assembly-specific decay heat limits is found with Eq. 2.20.

$$H_{t,(k)} - a_j.H(t_{HA,i})_{(k)} = 0 \textbf{ for } k = \{C_{cask}, ..., 2, 1\}, j \in vlrep[i] \tag{2.20}$$

Here, $H_{t,(k)}$ represents the transportation decay heat limit per assembly. Some cask designs use a uniform limit, and some are licensed for zoned loading, where each zone specifies the maximum decay heat per assembly in an area of the basket. Therefore, the comparisons are made by **order statistics**, where the hottest assembly in cask i at time $t_{HA,i}$ is constrained by the hottest position in the cask, the second hottest assembly is constrained by the second hottest position, and so on. This method allows for flexibility to handle either the flat or zoned limits since optimizing the placement of the assemblies inside each cask is beyond the scope of this research. Again, the time values for $t_{LC,i}$ and $t_{HT,i}$ are used first to evaluate the left hand side of each comparison, and if negative, the time required to reduce the decay heat to

the limit $H_{t,(k)}$ is found using linear interpolation.

The third objective incorporates the long-term goal of transportability into the solution search. By including it in the initial loading of the casks, it should aid in the future management of the fuel during decommissioning, which positively correlates with personnel dose, public exposure, and accident risks [142].

2.2.3 Constraints

The decision variables x, y, and t_{fill} are bound by a host of constraints to meet NRC regulatory requirements. The feasible region $\mathbf{\Omega}$ can be seen as the intersection of general categories of constraints, as shown in Eq. 2.21.

$$x \in \mathbf{\Omega} = \Omega_{bpp} \cap \Omega_{lc} \cap \Omega_{pool} \cap \Omega_{oper} \tag{2.21}$$

The subregions are bound by physical bin packing constraints (Ω_{bpp}), loading constraints (Ω_{lc}), spent fuel pool constraints (Ω_{pool}), and operational constraints (Ω_{oper}). The union of these subregions establish the safe and secure operation of an ISFSI. This section describes how each region was defined mathematically for the optimization.

2.2.3.1 Bin Packing Constraints

Two bin packing constraints were implemented to ensure physicality of the dry cask loadings. The first constraint given in Eq. 2.22(a) verifies that each assembly is only assigned to one cask, which is also known as the "no replacement" constraint.

$$\Omega_{bpp} = \begin{cases} \sum_{i=1}^{\overline{M}} x_{ij} = 1, \forall\ j \in \{1, ..., N\} & (a) \\ \sum_{j=1}^{N} x_{ij} \leq C_{cask} y_i, \forall\ i \in \{1, ..., \overline{M}\} & (b) \end{cases} \tag{2.22}$$

The second constraint Eq. 2.22(b) ensures that casks are not packed beyond their given capacity C_{cask}. When more than one cask type is used during the optimization, the variable C_{cask} will change based on which cask is used for bin i. The bin packing constraints used in this research are similar to the forms used in other bin packing studies [103, 130].

2.2.3.2 Cask Loading Constraints

Three cask loading constraints were implemented to ensure that the optimized selection of assemblies does not violate NRC regulatory requirements. These are shown in Eq. 2.23.

$$\Omega_{lc} = \begin{cases} a_j.BU \leq BU_{max}, \forall\ j \in \{1, ..., N\} & (a) \\ \sum_{j=1}^{N} x_{ij} \cdot a_j.H(t_{fill,i}) \leq H_{s,max}, \forall\ i \in \{1, ..., \overline{M}\} & (b) \\ \{a_j.H(t_{fill,i}) : x_{i,j} = 1\}_{(k)} \leq H_{s,(k)} \textbf{ for } k = \{C_{cask}, ..., 2, 1\}, & (c) \\ \qquad \forall\ i \in \{1, ..., \overline{M}\}, j \in \{1, ..., N\} \end{cases} \tag{2.23}$$

In the first constraint, the burnup BU of each assembly is compared to the maximum burnup BU_{max} specified in the CoC. The next two constraints ensure that the selected assemblies meet the decay heat limits for the cask and thereby protect its thermal performance. In Eq. 2.23(b), the combined heat load is constrained by the maximum heat load $H_{s,max}$ for storage listed by the CoC, and in (c), the individual decay heat values are compared to the per-assembly decay heat limits for storage $H_{s,(k)}$ using order statistics.

2.2.3.3 Pool Constraints

Two spent fuel pool constraints were implemented to ensure that the selected loading strategies would not negatively impact the operation of the pool. These

are shown in Eq. 2.24. The first constraint ensures that the power plant can be decommissioned in a timely fashion, and the second maintains the inventory of the pool below a given threshold.

$$\Omega_{pool} = \begin{cases} t_{fill,i} \leq t_{pool,close}, \forall\ i \in \{1, ..., \overline{M}\} & (a) \\ \sum_{j=1}^{N} x_{ij} \cdot W(i,j,t) \leq C_{pool}, \forall\ t \in \mathbb{R}^{+} & (b) \end{cases} \tag{2.24}$$

In Eq. 2.24(a), the variable $t_{pool,close}$ represents when the spent fuel pool is assumed to have completed decommissioning. For this research, it is assumed that the pool will close 5 years after the last projected discharge from all reactors feeding into it. The last discharge date is assumed to coincide with reactor retirement. For sites with multiple pools, the assemblies loaded into cask i are required to originate in the same pool, and $t_{pool,close}$ would correspond to that pool's shutdown date.

In Eq. 2.24(b), the variable C_{pool} represents the available capacity of the spent fuel pool. It is standard practice to leave enough space in spent fuel pools that they could accommodate the discharge of one full reactor core, known as the "Full Core Reserve" [135]. Therefore, the value for C_{pool} is set as the installed capacity minus the number of assemblies in one full core. The inventory of the pool is kept below this limit throughout the reactor lifetime and is confirmed by the mutliplication of x_{ij} with the time-dependent boolean $W(i,j,t)$, defined in Eq. 2.25.

$$W(i,j,t) = \begin{cases} 1 & : a_j.\text{discharge-date} \leq t < t_{fill,i} \\ 0 & : \text{otherwise} \end{cases} \tag{2.25}$$

Here, the boolean W returns 1 if assembly j is located in the pool at time t and 0 otherwise. A True value is assumed if t is located between the discharge date of

assembly j and the time at which it is loaded into cask i. The variable a_j retrieves the discharge date for assembly j from the library of assembly characteristics. The value of $W(i,j,t)$ is only computed when x_{ij} is equal to 1 to reduce the computational overhead, although the constraint is checked over the entire timeframe of the optimization. This is accomplished by evaluating Eq. 2.24(b) at every projected discharge date.

2.2.3.4 Operations Constraints

Two operational constraints were used to define the subregion Ω_{oper}. The first operational constraint limits the number of casks filled in a certain timeframe to a reasonable number. The second ensures that each cask is filled with enough used fuel assemblies to make the transfer worth performing for the utility. These constraints are formulated in Eq. 2.26.

$$\Omega_{oper} = \begin{cases} \sum_{i \in L_p} y_i \leq F_p, \ \forall \ p \in \{1, \cdots, n_p\} & (a) \\ \sum_{j=1}^{N} x_{ij} \geq \lfloor \eta(t) C_{cask} y_i \rfloor, \forall \ i \in \{1, ..., \overline{M}\} & (b) \end{cases} \tag{2.26}$$

In Eq. 2.26(a), the number of casks filled in every "loading period" p is less than or equal to the operational limit F_p. This reflects the operational reality that limits the number of casks that can be loaded in any given campaign. Utilities avoid transfer procedures during refueling outages, and sites with multiple pools usually only have one set of cask handling equipment [135]. For this research, F_p is assumed to be 10 casks per year unless the chosen site for optimization has a proven history of larger campaign sizes, in which case the maximum historical value is used. The optimization timeline is segmented into n_p periods from the beginning of the timeline to the end of decommissioning. Within each period p, the casks' y_i values are summed

over all of the casks belonging to the set L_p defined in Eq. 2.27.

$$L_p = \{i : t_p \leq t_{fill,i} < t_p + 1\text{yr}\} \tag{2.27}$$

In the second constraint, Eq. 2.26(b), the contents of each cask are compared to a minimum fill level $\eta(t)C_{cask}$. The time-dependent variable η is the fraction of the capacity that must be filled, shown in Eq. 2.28.

$$\eta(t) = \begin{cases} 1.0 & : t < t_{pool,decom} \\ 0.75 & : t \geq t_{pool,decom} \end{cases} \tag{2.28}$$

The variable $t_{pool,decom}$ is the date of the beginning of decommissioning for a pool and is assumed to coincide with the last projected discharge unless otherwise instructed. This is related to $t_{pool,close}$ by a difference of 5 years. Before decommissioning, the casks must be completely filled for a utility to spend resources on transfer procedures. During decommissioning, it is assumed that a higher priority would be given to closing the pool and that more resources would be available for transfer. Therefore, the minimum fill level is reduced to 75% to allow for more flexibility during this period.

2.2.4 Approach

A new metaheuristic algorithm is developed in this research to solve the dry cask loading problem. While its specifics are discussed in Section 3, Fig. 2.4 shows the basic modes of operation it shares with other successful memetic algorithms. The new algorithm uses a packing heuristic to assign assemblies to individual casks and thereby find new instances of the decision variable $\mathbf{x}$. The feasibility of new solutions is ensured through solution repair methods, and new nondominated solutions

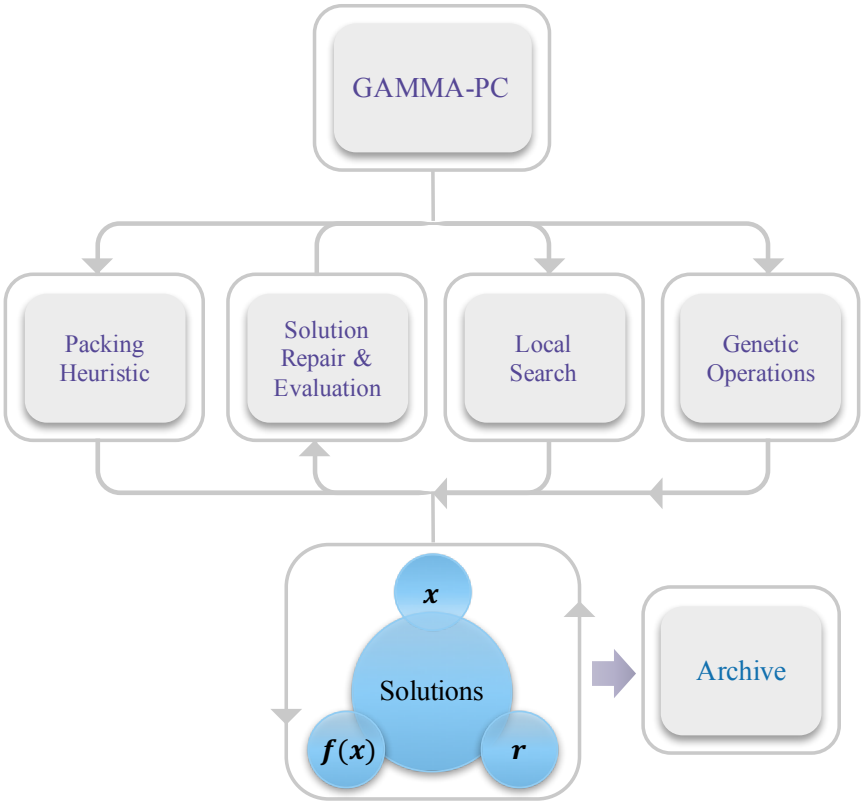


Figure 2.4: General optimization operators used to solve the dry cask loading problem. The details of the algorithm are discussed in Sec. 3.

are saved to an archive. During search phases, the new algorithm alternates between genetic operations and local search methods to balance the exploration and exploitation of the search space.

With this method, solutions are composed of three essential components: the decision variable $\mathbf{x}$, the objective vector $\mathbf{f}$, and residual tracking arrays $\mathbf{r}$. The latter is an addition to help guide the packing heuristic toward better solutions. For example, in [103] the residual matrix is a two-column array that tracks the remaining available weight and height for the bins to help the packing heuristic avoid overfilling them. In this research, $\mathbf{r}$ represents multiple arrays to guide the heuristic:

- r, a two column array tracking the open capacity and heat margin of each bin,

- *casksused*, an array identifying the dry cask type used for each bin,
- *poolpulled*, an array denoting the pool from which assemblies in each bin came,
- and *caskinfo*, a list containing a small dictionary of the available and filled positions by decay heat for each bin.

These tracking structures help guide the algorithm toward bins with available capacity and decay heat margins, while ensuring that items from the same pool are packed together.

The new method incorporates constraint handling in every phase except for the genetic operations, which only operate on the chromosome representation of the packing. The majority of the constraints are treated as "hard" constraints, meaning that solutions outside the feasible region are not explored. For example, assemblies are not packed into casks that cannot accomodate their decay heat at the time of transfer procedures, and a new bin can only be opened if the campaign has room to include it. However, the minimum fill requirement is treated as a "soft" constraint, and the heurisitics in the algorithm explore cask combinations without regard to maintaining this boundary.

While it could be evaluated simultaneously with the maximum capacity constraint during the solution evaluation step, the minimum fill constraint was handled separately to enable a more random exploration of the search space. The benefit of this separation is that the packing heuristic has more choices for where to place an assembly, allowing the algorithm to take advantage of the different strategies employed. If it were treated as a hard constraint, the overall flow of the algorithm would lean too heavily toward the First Fit strategy, which might favor one region of the objective space over another. Repair mechanisms return any infeasible solution back to the feasible region, which will be discussed more in Section 3.

3. DEVELOPMENT OF OPTIMIZATION ALGORITHM

Based on the complexities of the dry cask loading problem, a new metaheuristic algorithm was developed in this research, combining promising features to address various aspects of the dynamic bin packing problem. The new algorithm is similar to previous ensemble approaches for multiobjective bin packing problems. The general framework is based on NSGA-II [102] combined with local search techniques. A greedy randomized adaptive search procedure (GRASP) governs the packing and local search operators, based on previous research showing that GRASP-embedded approximate algorithms perform better for problems with special packing requirements [143]. The new ensemble is named the GRASP-enabled adaptive multiobjective memetic algorithm with partial clustering (GAMMA-PC). Section 3 presents a detailed view of GAMMA-PC and discusses how specific packing heuristics were adapted for the dry cask loading problem.

3.1 Algorithm Outline

As a population-based metaheuristic, GAMMA-PC maintains three sets of solutions throughout the computation: P, Q, and P_{EA}. The solutions in P represent those selected to be used as progenitors in the genetic operations, and the solutions in Q are their offspring as well as new solutions found by local search procedures. The set P_{EA} is the external archive of nondominated solutions and is updated every generation.

The general outline of GAMMA-PC is illustrated in Fig. 3.1. The algorithm is initialized by creating random packing solutions with the *GRASP-DBPP* operator. The GRASP wrapper is presented in Table 3.1 and contains various packing strategies for the dynamic bin packing problem (DBPP) within the *Construct-New-Solution*

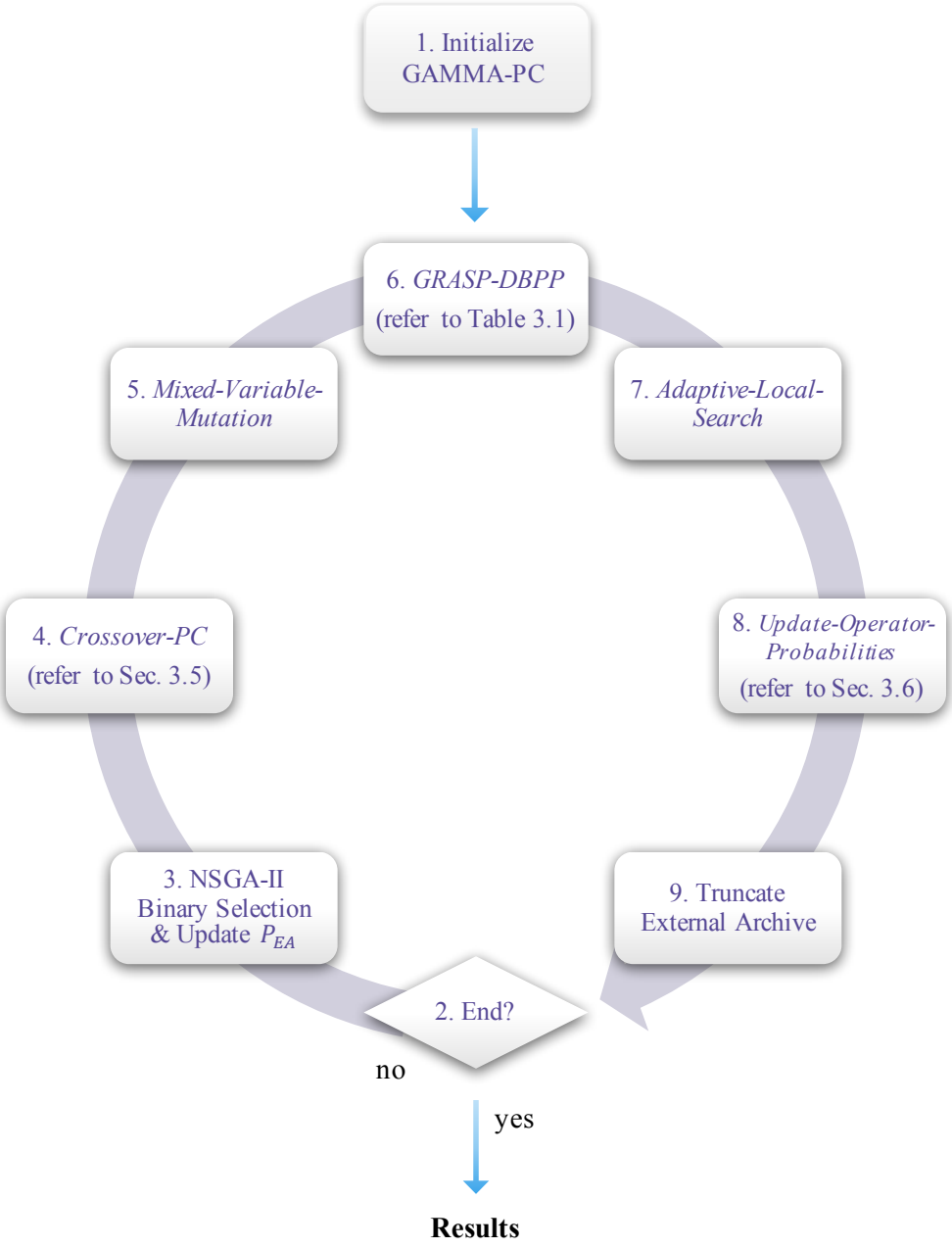


Figure 3.1: General schematic for the algorithmic flow of GAMMA-PC. The loop is repeated until the ending criteria is met.

Table 3.1: GRASP procedure for creating a new solution.

	Algorithm: GRASP-DBPP
	Input: $j = \{1, ...N\}$, *optional:* genetic information
	Output: Solution
1	$\mathbf{x}$:= Construct-New-Solution(input)
2	Create Solution from $\mathbf{x}$ and evaluate $\mathbf{F}$
3	Find a Neighbor using LocalSearch(Solution)
4	**return** BestSolution(Solution, Neighbor).

operator. The resulting decision vector $\mathbf{x}$ is then repaired and evaluated, which is discussed more in Sec. 3.4. A neighbor of the new solution is found, and then the best solution of the two is added to set Q. This procedure is repeated until the set contains enough solutions to fill the initial parent population.

While the GRASP wrapper is relatively unchanged from the greedy procedure in [144], the *Construct-New-Solution* operator is modified to address the dry cask loading problem. Its pseudocode is shown in Table 3.2. It can be called with or without genetic information, which consists of a real-valued chromosome representation of the items *chrom* and a list of suggested t_{fill} dates, t_{bank}. Items are assigned to bins in the order defined by *queue* based on a randomly chosen packing heuristic θ_b from a set of m options, where m represents the number of objective functions. The selected heuristic returns a restricted candidate list of bin options using a **cardinality** restriction, in which the list only contains the best β candidates [144]. Whenever a new bin is opened, a $t_{fill,i}$ value is chosen from a special continuous greedy function that is presented in Sec. 3.3. The packing layout is stored using variable length representation during the procedure, which is translated into the x loading matrix after all the items have been packed.

After creating a new solution, *GRASP-DBPP* finds one neighbor using a randomly selected local search operator. GAMMA-PC uses $m+1$ local search operators,

Table 3.2: Pseudocode for the GRASP-DBPP heuristic to construct new solution.

	Algorithm: Construct-New-Solution
	Input: items $j = \{1, ...N\}$
	optional: genetic information *chrom* & t_{bank}
	Output: *vlrep*, x, y, t_{fill}, **r**
1	Create empty y, t_{fill}, and **r** arrays of length $\overline{M}$
2	*queue* := *chrom* **if** *chrom* else *range*(N)
3	RCL_t = InitializeGreedyFunction()
4	Select θ_b at random from range(m)
5	**for** $j \in$ *queue* **do**
6	Construct RCL_i based on mode θ_b
7	Select i at random from RCL_i
8	**if** $y_i = 0$:
9	Append [j] to *vlrep*
10	Select θ_t at random from range($m - 1$)
11	Construct RCL_t based on mode θ_t
12	Select $t_{fill,i}$ from RCL_t or from t_{bank}
13	AdaptGreedyFunction(t_{fill})
14	$y_i = 1$
15	**else**:
16	Append j to $vlrep_i$
17	AdaptGreedyFunction(*vlrep*)
18	**end if**
19	**end for**
20	Construct x based on *vlrep*

where each operator explores a carefully selected local neighborhood to minimize one of the objective functions. Two operators represent the first objective of bin packing: minimizing the number of bins. One search operator moves a solution toward having fewer bins, while the other repacks bins to find new nondominated solutions. The latter is chosen if a solution has already reached the theoretical minimum number of bins.

The parent population P is selected every generation using NSGA-II-styled binary selection. Sets P and Q are combined and sent through the *fast-non-dominated-sort* procedure described in [102]. Then, the top solutions by nondomination levels and crowding distance assignments are sent to the *Crowded-Comparison-Operator* to select a new set P, as described in Sec. 2.1.2.

In Steps 4 and 5 of GAMMA-PC, the solutions in P are sent through the genetic operators of *Crossover-PC* and *Mixed-Variable-Genetic-Mutation*. The genetic operations in both are modified from their traditional form to account for the mixed-variable environment. Both steps handle the chromosome representation of the packing and the t_{fill} decision variable separately. The crossover operator performs separate single-point crossovers on the chromosome representation and on the t_{fill} array. The modification to the genetic mutation operator is more complicated. The mutation first performs a two-point swap in the chromosome representation as it would before mixed-variables were introduced. Then, the $\mathbf{t}_{fill}$ array is modified using a new technique for a bin packing problem.

Previous research with mixed-variable optimization has advised the mutation of a continuous variable using a normal distribution based on the variability present for that characteristic in the population members [145]. The function to perform this mutation, translated into the particulars of the dynamic problem, is shown in

Eq. 3.1.

$$t_{fill,i,g+1} = t_{fill,i,g} + \mathcal{N}(0, \sigma_{i,g}),\ \forall\ i \in \{1, ..., \overline{M}\} \tag{3.1}$$

Here, g represents the generation, and $\sigma_{i,g}$ is the standard deviation of the values for $t_{fill,i}$. Because the dynamic bin packing problem uses a variable number of bins and the bins are not necessarily sorted by their fill time, performing the mutation based on the bin number i can lead to errors. To ensure that the mutation produces a usable $\mathbf{t}_{fill}$ matrix, the function was modified for this research to the form given in Eq. 3.2.

$$t_{fill,i,g+1} = t_{fill,i,g} + \mathcal{N}(0, \sigma_{cat}),\ \forall\ i \in \{1, ..., \overline{M}\} \tag{3.2}$$

Here, the normal distribution is based on the standard deviation of time values within a particular category. To find these values, all of the fill times used by the solutions in set P are filtered and analyzed based on the *Bin-Packing-Time-Sequencing* algorithm, shown in Table 3.3. The values are grouped together within the specified interval range Δ_t. This algorithm returns a list T_{sd} of standard deviations along with the temporal boundaries for each category so that when mutation occurs, each $t_{fill,i}$ can be mutated with the standard deviation in its own time category. The new genetic information is then sent to *GRASP-DBPP* in Step 6 to be decoded into real solutions.

In Step 7 of GAMMA-PC, local search is performed on the new solutions in Q with probabilities that are updated every generation. The same $m + 1$ local search operations used in *GRASP-DBPP* are called here, searching for N_{LS} neighbors instead of only one. On even generation numbers, the solutions in P_{EA} are clustered into groups based on the number of bins used, and a random solution in each cluster is sent to a randomly chosen local search operator. The solutions at the extremes are always included, searching near one of the solutions in P_{EA} with an ideal value

Table 3.3: Pseudocode of the *Bin-Packing-Time-Sequencing* algorithm to produce categorical standard deviations for mutation of $\mathbf{t}_{fill}$ matrices.

	Algorithm: Bin-Packing-Time-Sequencing
	Input: Δ_t, $S = \{t_{fill,i} \neq 0.0,\ \forall\ i \in \{1, ..., \overline{M}\}$, $\mathbf{t}_{fill} \in \{\mathbf{t}^1_{fill}, \mathbf{t}^2_{fill}, ..., \mathbf{t}^{pop}_{fill}\}_g\}$
	Ouput: T_{sd}
1	$t_{min} = t_{low}$ = minimum time in S
2	t_{max} = maximum time in S
3	$n_{cat} = (t_{max}$ - $t_{min})/\Delta_t$
5	$t_{high} = t_{low} + \Delta_t$
6	$T_{sd} := \varnothing$
7	**for** *cat* in range(n_{cat}) **do**
8	$t_{cat} = \{t \in S : t_{low} \leq t < t_{high}\}$
9	$\sigma_{cat} \leftarrow$ Calculate the standard deviation of t_{cat}
10	Add (t_{low}, t_{high}, σ_{cat}) to T_{sd}
11	$t_{low} = t_{high}$
12	t_{high} += Δ_t
13	**end**

in their fitness vector. Finally, every four generations, local search is performed across P_{EA} according to the same probabilities used for the local search of Q. Any neighbor found to be nondominated to the input solution during local search is added to Q. After every n_{trc} generations, P_{EA} is truncated based on the crowded distance assignment value to keep the size of the archive below a preset level.

3.2 Packing Heuristics

During the creation of a new solution, four main packing heuristics are employed:

- the *Maximum Move* strategy,
- the *Least Loaded* strategy,
- the *Dot Product* strategy,
- and the *First Fit-T* strategy.

These approximate algorithms are employed to guide the packing process toward better regions of the objective space than a random loading alone. However, with

Table 3.4: Pseudocode to construct RCL_i for the *Construct-New-Solution* algorithm.

```
    Algorithm: Construct RCL_i
    Input: r, j, θ_b, β
    Ouput: RCL_i
1   bins ← Sort the bin indices using r and the priority set by θ_b
2   RCL_i := ∅
3   for i in bins do:
4       if packable(i, j):
5           Add i to RCL_i
6           if |RCL_i| = β:
7               break
8   end
9   if includenewbin(j, θ_b):
10      Add i = |bins| + 1 to RCL_i
```

the exception of the *First Fit-T* strategy, some randomization is incorporated in each procedure through the use of restricted candidated lists to promote diversity.

The first three packing strategies follow the template set in Table 3.4. The bin indices are sorted using the residual matrix r and the priority set by mode θ_b. With the *Maximum Move* strategy, the order is set from the most filled to the emptiest by capacity, and with the *Least Loaded* strategy, it is set from the lowest to the highest heat load to find better distributions of decay heat. Under the *Dot Product* strategy, the order is set from the highest to the lowest value of the metric in Eq. 3.3, which was modified from [103].

$$dp_i = C_{cask} \cdot r_{i,0} + H_{s,max} \cdot r_{i,1} \tag{3.3}$$

In this equation, the residual capacity for bin i is multiplied by the cask capacity, and the residual heat load is multiplied by the cask thermal limit. By including both, the largest dp_i value balances the available physical space with the available heat capacity.

The restricted candidate list is composed of the first β bins that would be able

to accomodate item j as determined by the *packable* boolean. With the dry cask loading problem, *packable* determines if assembly j was located in the spent fuel pool when cask i was loaded, if it would be cool enough to meet storage thermal limits, and if cask i had a available spot to store the assembly. After constructing RCL_i, the boolean *includenewbin* is evaluated to determine if opening a new bin should be available as an option. This value is True if RCL_i is empty and the number of bins is below $\overline{M}$ or if the number of open bins is below a given threshold. However, under the *Maximum Move* strategy, it is only True when the restricted candidate list is empty to encourage a lower number of bins.

The *First Fit-T* strategy modifies the template in Table 3.4 by restricting RCL_i to the first bin in which item j is *packable*. The bins are evaluated in chronological order of their $t_{fill,i}$ values to encourage earlier loading times and thereby alleviate thermal stresses in the spent fuel pool. The procedural loop is also modified to both check if bin i can accept item j and, if not, to check if $t_{fill,i}$ could be modified so that item j would be *packable*. The latter evaluation verifies that the new $t_{fill,i}$ value would not adversely affect the items already packed in bin i before making the change and returning i. If the change would cause any item to be too hot or unavailable at the new loading time, bin i is passed, and the next bin is considered. New $t_{fill,i}$ values are generated using the continuous greedy function RCL_t.

3.3 Continuous Greedy Function for Selecting Fill Times

The continuous greedy function to generate $t_{fill,i}$ values combines the idea of Monte Carlo selection with that of a restricted candidate list. The function RCL_t used in *GRASP-DBPP* (see Table 3.2) maintains the timeline and keeps track of the available space in the intermediate holding area, i.e. the spent fuel pool, as a function of time. It also keeps track of the number of bins filled within each period.

Table 3.5: Algorithm to initialize t_{fill} restricted candidate list.

	Algorithm: InitializeGreedyFunction
	Input: F_p, C_{pool}, t_{close}, $\{(t_a, n_{batch,a}) \; \forall \; a \in \{1, ..., b_n\}\}$,
	optional: vlrep, t_{fill}
	Ouput: RCL_t
1	$t_{range} = \{t_a\} \; for \; a \in \{1, ..., b_n\}$
2	$space = \{s_a = s_{a-1} - n_{batch,a}\}$ for $a \in \{1, ..., b_n\}$ with $s_0 = C_{pool}$
3	Append t_{close} to t_{range}, s_{b_n} to *space*
4	$periods = \{year(t_1), ..., year(t_{close})\}$
5	$r_{fill} = \{F_p\}$ for $p \in periods$
6	**if** *optional* arguments given:
7	**for** every bin i in *vlrep* **do**
8	AdaptGreedyFunction($vlrep[i]$, $t_{fill,i}$)

When the function is called upon to generate a new fill time, RCL_t restricts the timeline and returns a randomly selected time from within that range.

Table 3.5 presents the algorithm used to initialize RCL_t. The function begins as a list of arrival times associated with batches of items. The t_a values in the list plus the t_{close} date define the timeline of possible $t_{fill,i}$ values. The *space* list keeps track of the available space in the holding area, which is reduced by $n_{batch,a}$ at t_a. The r_{fill} list keeps track of the number of bins filled per period and is initialized at full capacity F_p. If RCL_t is initialized from an existing solution, the variable length representation and the t_{fill} array are used to adapt the greedy function, increasing the *space* values to match the removal of items from the holding area and reducing the r_{fill} values for every bin filled.

When RCL_t is used to generate a new fill time, the t_{range} and *space* arrays are converted into a probability density function. While the *space* list is allowed to hold negative values, these are converted to 0% probability in the probability density function. The probability density function in turn defines a cumulative density function. A new $t_{fill,i}$ value is chosen by randomly selecting a percentile within a given range

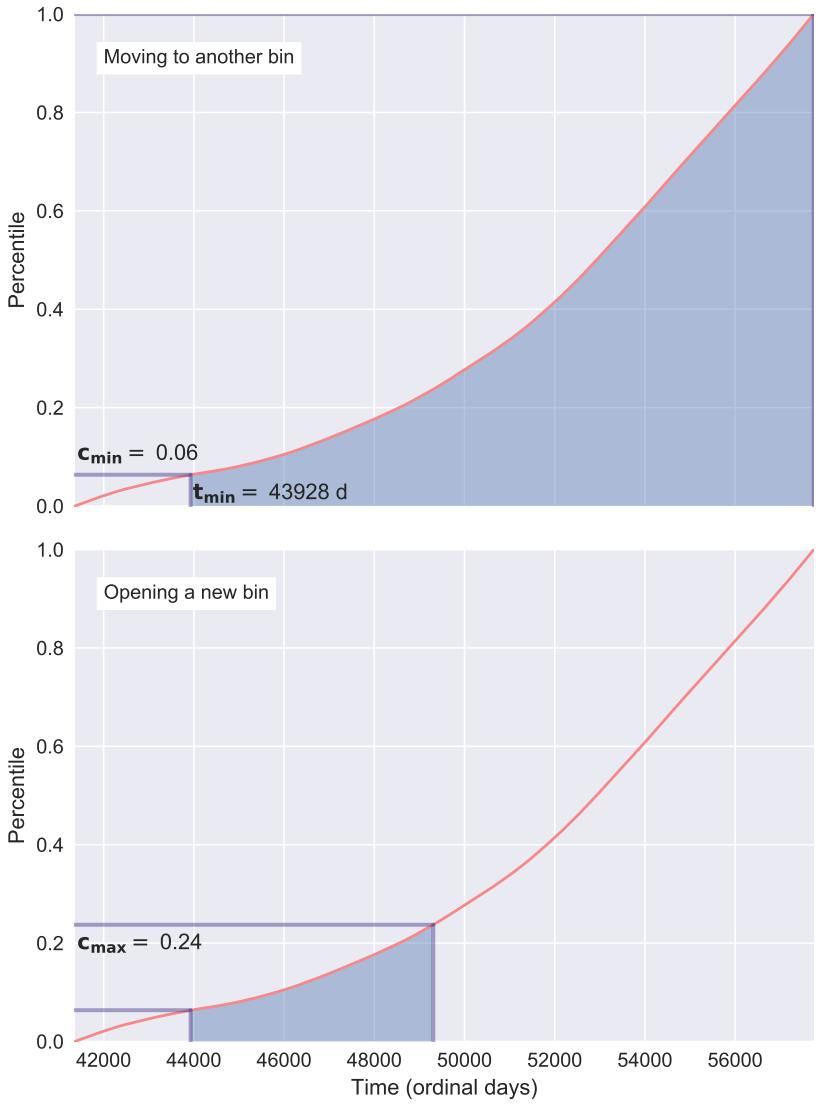


Figure 3.2: Illustration of the continuous greedy function. The top plot presents an example of how the greedy function restricts the timeline when moving an assembly from one cask to another, while the bottom presents the restriction for opening a new cask.

and inserting it into the inverse cumulative density function.

Fig. 3.2 presents two different examples of how the cumulative density function operates for the dry cask loading problem. The timeline is first restricted by setting a minimum value at the arrival time of the item under consideration. To move an item from one bin to another, the maximum time value in the plot is set at the end of the timeline, although if the spent fuel pool were ever filled beyond capacity, t_{max} would be set to the time when the overflow occured. Corresponding percentiles are found for the minimum and maximum time values, c_{min} and c_{max} respectively, and a random percentile is chosen within that range. The chosen percentile is then converted back into a new time value using an inverse cumulative density function.

There are two different continuous greedy functions shown in Fig. 3.2 to emphasize that the new time generator for RCL_t can be adjusted to the problem at hand. The top figure shows the restriction of the cumulative density function when moving an item, and the bottom figure shows the restriction when opening a new bin with an item. The restriction adapts to the type of movement. In the top figure, the greedy function restricts the timeline based only on the arrival time of the item to the spent fuel pool and space constraints in the pool. The corresponding c_{max} is set at 1.0. In the bottom, it also restricts the timeline based on available positions in loading campaigns, resulting in a much lower c_{max}. In both cases, c_{min} is set to correspond to the discharge date of the assembly from the reactor.

3.4 Solution Repair Methods

After a new solution is decoded in *GRASP-DBPP* or found through local search methods, it undergoes a three step process before being added to set Q, as shown in Fig. 3.3. In the first step, the solution is evaluated to determine if it is in the feasible region or not. If the solution violates a soft constraint, it is repaired in Step 2 before

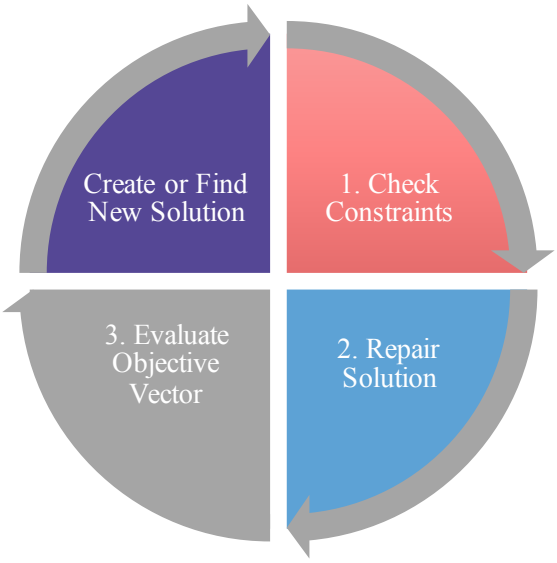


Figure 3.3: Steps to check and evaluate a new solution.

having its objective vector evaluated in Step 3. The first and the last steps in this process happen according to the mathematical paradigm laid out in Section 2.2. The intermediate step is discussed here.

Soft constraint violations are repaired in GAMMA-PC using four main techniques:

- *Add-to-bin*,
- *Empty-bin*,
- *Fill-bin-later*,
- and *Fill-bin-earlier*.

The first two methods move items from one bin to another, and the second two focus on moving bins along the optimization timeline. During the repair of a solution, an action is called based on the characteristics of the solution with some randomness in

Table 3.6: Pseudocode for the greedy *Add-to-bin* technique.

	Algorithm: Add-to-bin
	Input: i, r_{goal}, Solution
	Ouput: Solution
1	Make RCL_{pull}
2	Retrieve $vlrep$ and r from Solution
3	**for** k in $range(c)$ **do**:
4	Select i_2 at random from RCL_{pull}
5	Select random j from $vlrep[i_2]$ that is $packable(i, j)$
6	**if** j found:
7	Move j from i_2 to i
8	**break if** $r_{i,0} = r_{goal}$
9	Remove i_2 from RCL_{pull} **if** j could not be found or $r_{i_2,0}$ meets the minimum fill requirement during decommissioning
10	Refill RCL_{pull} if needed

the method selection. For example, if one of the bins i is too empty before decommissioning of the pool but meets the minimum fill limit during the decommissioning period, *Fill-bin-later* is selected. However, if it is too empty in either period, *Add-to-bin* or *Empty-bin* is selected randomly instead. The probability of *Empty-bin* being chosen increases for emptier bins and decreases for bins with more items in them. Finally, *Fill-bin-earlier* is employed if the loading campaigns during decommissioning are full, and some of the bins loaded during that period are filled to capacity.

The *Add-to-bin* and *Empty-bin* techniques incorporate restricted candidate lists as a way to increase the likelihood of making better decisions. The pseudocode for *Add-to-bin* is shown in Table 3.6. The restricted candidate list RCL_{pull} is formed by prioritizing bins that either can feasibly be filled during the decommissioning period or also violate the minimum fill constraint. When these types of bins contribute items to the target bin i, the movements either do no harm or speed up the process of bringing the solution back into the feasible region. The *Empty-bin* technique uses a procedure similar to the *First Fit-T* strategy to remove items to other bins and forms the RCL_i list for each item based on the dot product evaluation in Eq. 3.3.

The *Fill-bin-later* and *Fill-bin-earlier* techniques use the continuous greedy time function RCL_t to find new $t_{fill,i}$ values. Before making a change to the loading time, the *binadjustable* boolean is evaluated. This boolean determines if the time change would negatively impact any of the items in the bin or the inventory in the spent fuel pool, and if it would, the boolean is set to False. The $t_{fill,i}$ value is only changed if *binadjustable* is shown to be True.

The repair techniques are performed on the solution until all of its bins have been brought back into the feasible region. In the process, they also move the solution toward one region of the objective space or another. The *Empty-bin* technique reduces the number of open bins, thereby moving the solution closer to the ideal value for the first objective function. The *Add-to-bin* technique preferentially pulls from later-filled bins, which tend to contain hotter assemblies from later future core discharge projections. This can move the bins as a whole toward lower average initial heat loads or lower cask-average dose rates. Therefore, these techniques are also included in the local search methods with a slightly different emphasis depending on the focus.

3.5 Crossover with Partial Clustering

The standard genetic crossover used in NSGA-II is modified in GAMMA-PC to include partial clustering with the goal of balancing exploration and exploitation of the objective space. The pseudocode for the *Crossover-PC* operator is shown in Table 3.7. It first sends a fraction f_{ran} of P straight to the crossover operator to be randomly paired and mixed. Then, it carefully sorts the remaining solutions into N_c clusters before sending each cluster to the crossover operator. In this research, N_c was set at $2m$. All of the solutions produced by the multiple crossovers are combined to make up Q.

Table 3.7: Crossover operator with partial clustering.

	Algorithm: Crossover-PC
	Input: P, f_{ran}
	Ouput: Q
1	Split P into P_{ran} and P_c based on f_{ran}
2	$Q_{ran} \leftarrow$ Mixed-Variable-Genetic-Crossover(P_{ran})
3	Update ideal values z^* using P_{EA}
4	Sort solutions from P_c into N_c Tchebycheff clusters
5	$Q_c := \emptyset$
6	**for** c in range(N_c) **do**
7	**if** cluster c is not empty:
8	New solutions $\leftarrow$ Mixed-Variable-Genetic-Crossover(cluster c)
9	Add New solutions to Q_c
10	**else:**
11	Update-Mutation-Rate
12	**end if**
13	**end for**
14	$Q = Q_{ran} + Q_c$

Crossover-PC performs the grouping of the clusters using the Tchebycheff approach [146]. To support the use of this approach without knowledge of the Pareto front, the ideal values z^* are updated every generation to reflect the most ideal points found in the objective space for every objective θ. The $m \times 1$ array z^* is set using (3.4).

$$z^*_\theta = \min_{\mathbf{x} \in \Omega} f_\theta(\mathbf{x}) \tag{3.4}$$

To perform the sorting, N_c random weight vectors of the form $\lambda = (\lambda_1, ..., \lambda_m)^T$ are produced such that $\sum_{\theta=1}^{m} \lambda_\theta = 1$ and $\lambda_\theta \geq 0$ for all $\theta = 1, ..., m$ [146]. Then, the objective vectors for each solution in P_c are transformed into single objective fitness values through (3.5) with the random weight vectors.

$$g^{te}(\mathbf{x}|\lambda) = \max_{1 \leq \theta \leq m} \left\{ \frac{\lambda_\theta}{w_\theta} (f_\theta(\mathbf{x}) - z^*_\theta) \right\} \tag{3.5}$$

This procedure is similar to the process of decomposition in MOEA/D-type algo-

rithms but is only applied during the crossover operation.

The form of (3.5) is also modified from the Tchebycheff approach generally used by MOEA/D [146] to reflect the purpose of the weight in the Tchebycheff norm, which is to normalize various criteria [147]. The λ weight vector is still included in (3.5), but it is divided by the vector w. This is calculated in similar manner as z^* in (3.4), except finding the maximum value present for each objective among the solutions in P_{EA}. The inclusion of w ensures that the clusters are formed throughout the objective space, even if one objective function explores a much larger range than the others. After the single objective fitness values are calculated for a solution, it is placed in the cluster with the weight vector resulting in the smallest single objective value.

3.6 Updating Operator Probabilities

The first "A" in GAMMA-PC refers to the adaptation performed every generation to move the operator probabilities toward areas of need during calculations. The first change is made during *Crossover-PC* in Step 4. For every empty cluster found, the mutation rate increases by a small amount. This change occurs because the single objective fitness values represent different areas of the objective space, so if a cluster is empty, the area governed by that weight vector has not been explored well. The increase is made in the mutation rate to encourage more random exploration.

The local search probabilities are also updated every generation to encourage search in one area or another based on the solutions present in P_{EA}. GAMMA-PC maintains m local search probabilities: the probability of local search overall p_{ls} and the probabilities of objective-specific local search operators $p_{ls,\theta}$ for the first $m-1$ objectives. The probability of local search for the last objective is implicit because the total probablity of the objective-specific operators sums to 1.0.

Table 3.8: Algorithm to update local search probabilites.

	Algorithm: Update-Operator-Probabilities
	Input: P_{EA}, p_{ls}^{g}, $p_{ls,\theta}^{g}$ for θ = 1,...,$m-1$
	Ouput: p_{ls}^{g+1}, $p_{ls,\theta}^{g+1}$ for θ = 1,...,$m-1$
1	$N_{size}^{*} := m \times 1$ array of zeros
2	**for** θ in range(m) **do**
3	$\quad IdealNeighbors = \{u \in P_{EA} : u_\theta - z_\theta^{*} \leq 0.10(z_{max,\theta} - z_\theta^{*})\}$
4	$\quad N_{size,\theta}^{*} = \lvert IdealNeighbors \rvert$
5	**end for**
6	$p_{ls}^{g+1} \leftarrow$ Update-Local-Search-Rate(p_{ls}^{g}, N_{size}^{*})
7	$p_{ls,\theta}^{g+1} \leftarrow$ Redistribute-Sub-Local-Search-Rates($p_{ls,\theta}^{g}$, N_{size}^{*})

The algorithm to update the local search probablities is given in Table 3.8. First, the number of solutions in P_{EA} in the local neighborhood of each ideal value are counted, found using the relationship shown in Step 3 of *Update-Operator-Probabilities*. The size of each neighborhood is stored in the $m \times 1$ array N_{size}^{*}. Then, the sizes of the ideal neighborhoods are used to update the local search probabilities.

The N_{size}^{*} array is sent to the *Update-Local-Search-Rate* operator first, which adjusts the overall probability p_{ls}. The operator increases p_{ls} if any value in N_{size}^{*} falls below a preset minimum value or decreases it if all of the values are above a preset maximum, stopping at a minimum boundary of $p_{ls} = 0.1$. The preset minimum and maximum neighborhood size values were set at 5 and 20 in this research, respectively.

Next, the probabilities for the individual local search operators are updated in *Redistribute-Sub-Local-Search-Rates* based on the variation in N_{size}^{*}. The probability of the local search operator for the objective with the largest ideal neighborhood is reduced by a fraction determined by the size of the smallest ideal neighborhood. The portion it loses p_{move} is redistributed fairly among the probabilities for the other objectives. The value for p_{move} is calculated using Eq. 3.6.

$$p_{move} = \left(1 - \frac{\min(N_{size}^{*})}{\max(N_{size}^{*})}\right) p_{ls,\theta(\max)}^{g} \tag{3.6}$$

Here, $p^g_{ls,\theta(\max)}$ is the probability of local search using the operator for the objective with the largest ideal neighborhood. The portion p_{move} is split and added to the probabilities for the other operators based on their ideal neighborhood sizes.

For example, if the probabilities at the end of generation g were $p^g_{ls,1} = 0.25$, $p^g_{ls,2} = 0.25$, and $p^g_{ls,3} = 0.50$ and if N^*_{size} were found to be $(2, 4, 8)$, then $p_{ls,3}$ would lose $(1 - 2/8)0.50$ from its share, or $p_{move} = 0.375$. A fair way to redistribute this between the search operators for the first and second objectives would be to give $p_{ls,1}$ a larger chunk of p_{move} since its neighborhood is smaller. Therefore, the probabilities would be updated to $p^{g+1}_{ls,1} = 0.50$, $p^{g+1}_{ls,2} = 0.375$, and $p^{g+1}_{ls,3} = 0.125$ in the next generation. Whatever redistribution scheme is used, though, it must be ensured that $\sum_{\theta=1}^{m} p^{g+1}_{ls,\theta} = 1$. Moreover, the redistribution only takes place if $p^g_{ls,\theta(\max)}$ is above 0.1 to prevent the removal of one type of local search altogether.

3.7 Connecting GAMMA-PC with UNF-ST&DARDS

GAMMA-PC was implemented in a Python package for demonstration and distribution. It was designed to be adaptable, maintainable, extensible, and modular, and its structure is shown in Fig. 3.4. Each class within the package includes modules to support its technical function. For example, the assembly class includes decay heat and dose modules, and the evolutionary framework includes modules to maintain the external archive and to update the operator probabilities. The package also includes unit testing modules for each class to ensure that new changes to either the multiobjective problem or the algorithmic structure of GAMMA-PC do not cause errors in the rest of the code.

GAMMA-PC draws information from the UNF-ST&DARDS database during its initialization phase. Figure 3.5 lists the information that is saved. GAMMA-PC pulls data about the site being optimized, the dry cask type(s) being used, and the current

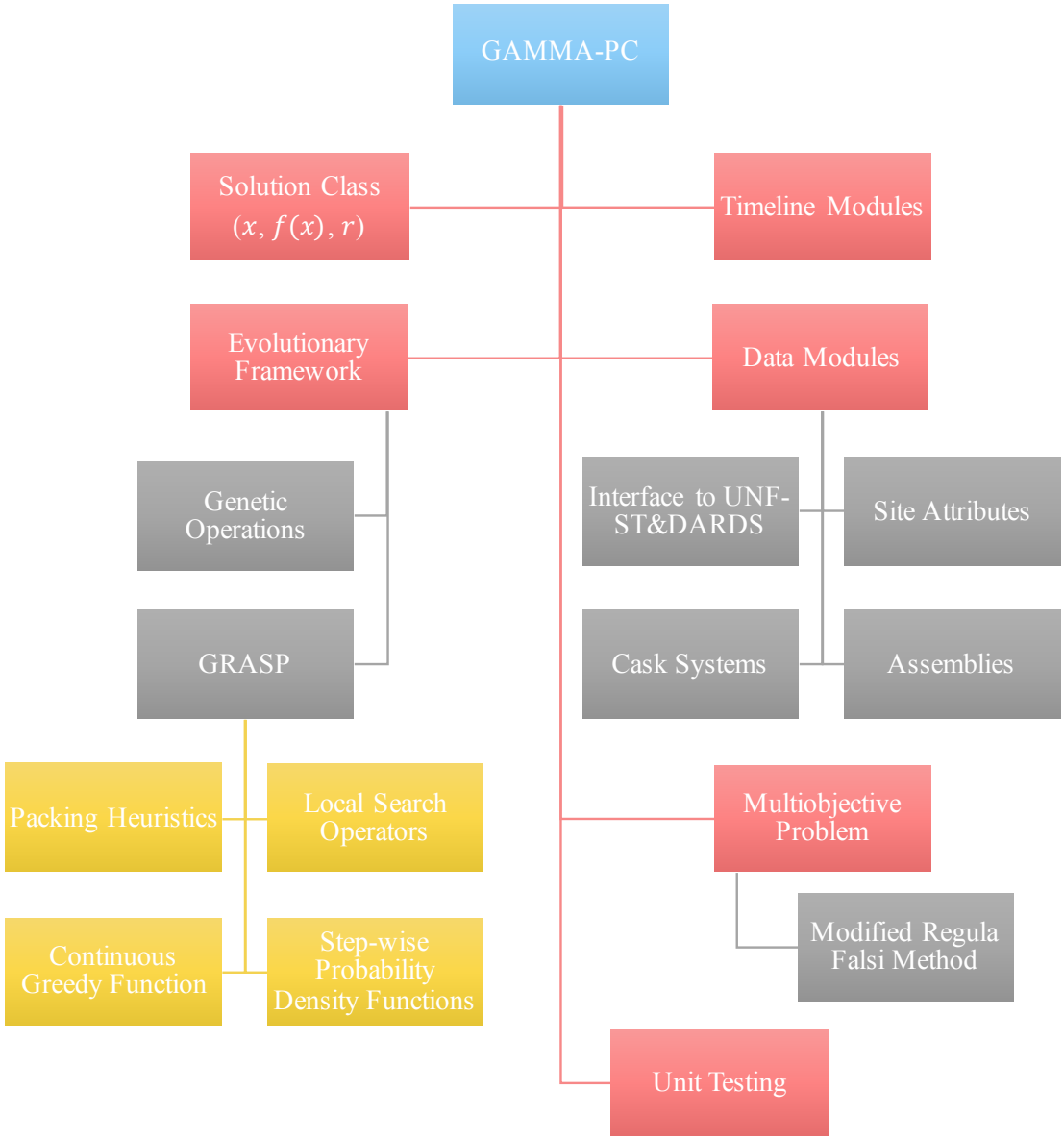


Figure 3.4: Schematic of the modular architecture of the Python package for GAMMA-PC.

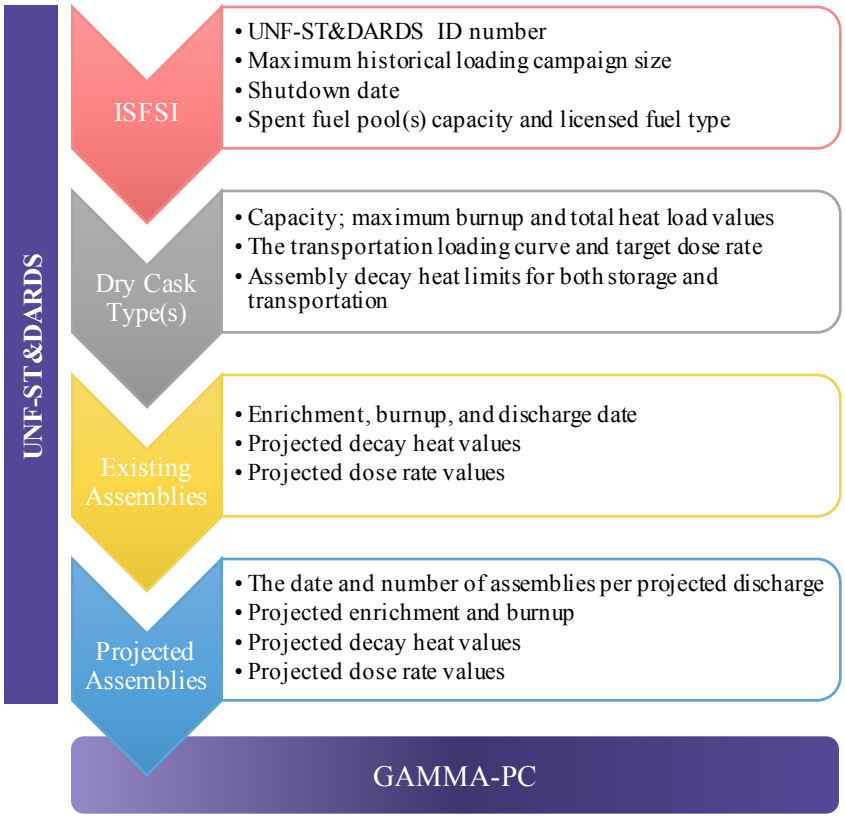


Figure 3.5: Information pulled from UNF-ST&DARDS.

and future assemblies in the spent fuel pool(s). This data is saved by the algorithm into the correct variable assignments and is used throughout the calculations as described in Section 2.2.

A few assumptions are made while the problem is constructed. The first is that the spent fuel pool(s) begins decommissioning in conjunction with the last projected discharge to the pool. The value saved for the number of casks loaded in a period F_p is also assumed to be either the maximum historical campaign size or 10 casks, whichever is greater. Finally, the value saved for the pool capacity C_{pool} is set to be one reactor core size smaller than the value imported from UNF-ST&DARDS.

The methodological performance of GAMMA-PC was evaluated during the course

of this research and is discussed in Appendices A and B. Through this analysis, it was shown that GAMMA-PC performs as well as other metaheuristic algorithms at solving the dynamic bin packing problem. It also produced more diverse solutions that better approximated the best-known Pareto front for a small example problem. To further demonstrate its performance, Section 4 describes the application of GAMMA-PC to the dry cask loading problem for a couple of different ISFSIs and analyzes the results.

4. DEMONSTRATION CASES

GAMMA-PC was designed to be a robust and adaptable multiobjective metaheuristic specially made to handle the complexities of the dry cask loading problem outlined in Sec. 2. To demonstrate its performance in this capacity, the new algorithm was applied to ISFSIs at Vermont Yankee, Comanche Peak, and Zion nuclear power plants. These three sites reflect the diversity of the broader U.S. nuclear fleet. Their algorithmic parameters are shown in Table 4.1. This section presents the results of applying GAMMA-PC to each case and evaluates its performance as a loading metaheuristic.

4.1 Vermont Yankee

The Vermont Yankee Nuclear Power Station was a single BWR that began operation in 1972 and was permanently shutdown in 2014 [148]. Transfer procedures to remove assemblies from the spent fuel pool commenced in the summer of 2017 and are expected to finish in 2018 [149]. The used fuel transfers are being conducted using ALARA strategies, but further information about the loading objectives is currently unavailable.

GAMMA-PC performed the optimization of the Vermont Yankee assembly selection according to the paradigm developed in Sec. 2. The number of assemblies

Table 4.1: Nuclear power plant characteristics for the demonstration cases.

Site	Reactors	N	GAMMA-PC t_0	$t_{pool,close}$	F_p
Comanche Peak	2 PWRs	7239	2013 Mar. 30	2058 Feb. 2	10 casks/year
Vermont Yankee	1 BWR	2993	2017 July 1	2018 Dec. 30	5 casks/month
Zion	2 PWRs	2226	2014 Jan. 1	2015 Jan. 31	7 casks/month
			2005 Jan. 31	2015 Jan. 31	10 casks/year

Table 4.2: Dry system characteristics for the Vermont Yankee Nuclear Power Plant.

	Vermont Yankee
Cask System	HI-STORM 100 MPC-68
C_{cask}	68
$H_{s,max}$ [kW]	34.0
$H_{t,max}$ [kW]	18.5
$BU_{s,max}$ [GWd/MTU]	60.0
No. of Casks Needed:	
Lower Bound	45
Maximum $\overline{M}$	59
Function Evaluations	15,000

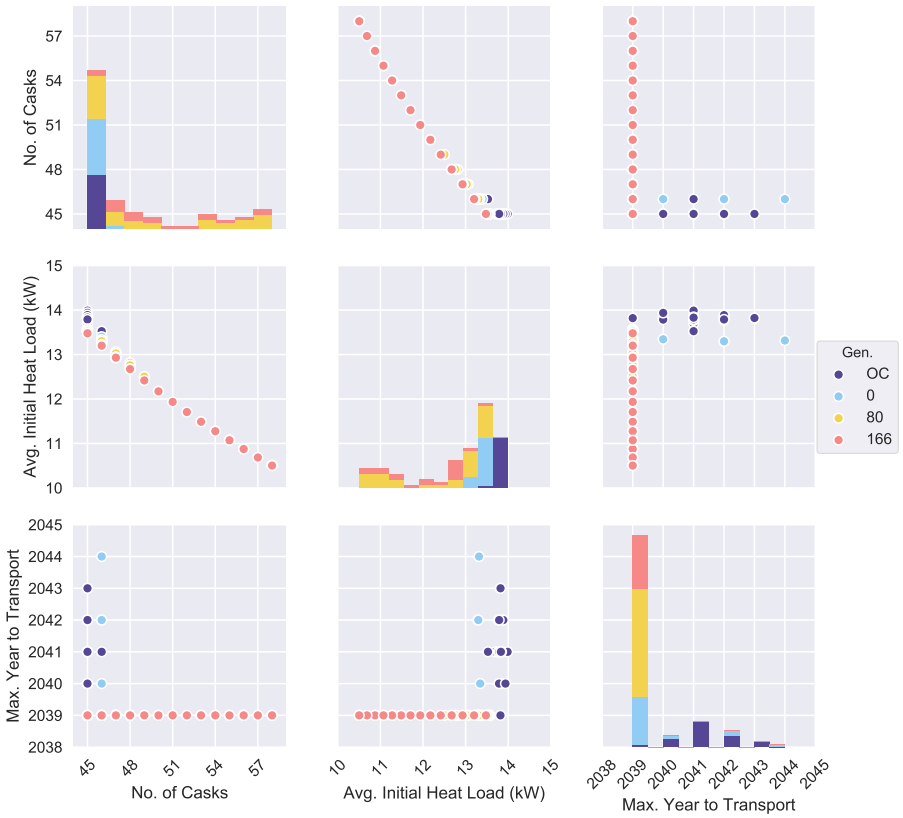


Figure 4.1: Scatter matrix showing objective vectors for GAMMA-PC and for OC solutions in the Vermont Yankee dry cask loading problem. This figure plots the evolution of the solutions found by GAMMA-PC with the initial generation (0), an intermediate generation (80), and the approximate set, found in generation 166. Only the top 40% best solutions are depicted from sets OC, 0, and 80.

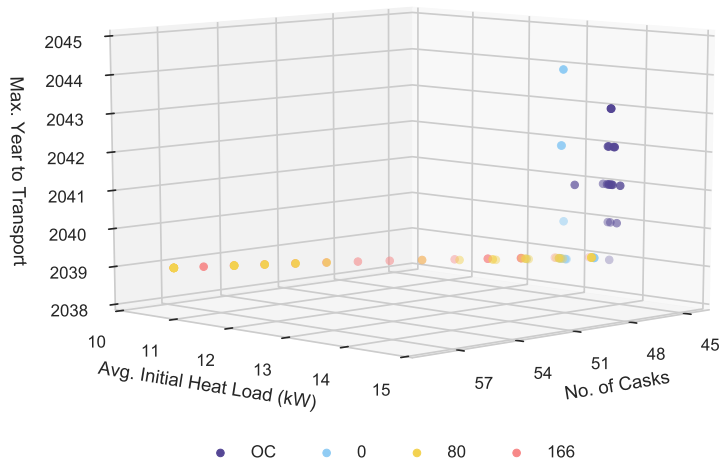


Figure 4.2: Scatter plot showing objective vectors for GAMMA-PC and OC solutions in the Vermont Yankee dry cask loading problem. This figure plots the evolution of solutions found by GAMMA-PC with the initial generation (0), an intermediate generation (80), and the approximate set, found in generation 166. Only the top 40% best solutions are depicted from sets OC, 0, and 80.

N included in the packing problem was limited to the fuel present in the spent fuel pool at the beginning of 2017. One adjustment was made to the F_p limit defined in Eq. 2.26 to ensure the completion of transfer procedures within the small timeframe shown in Table 4.1. The "bin" used to pack the assemblies was the HI-STORM 100, MPC-68 [149], whose characteristics are given in Table 4.2. The minimum number of casks to store the fuel is 45, and the upper bound for GAMMA-PC was set at 59 casks based on the decommissioning minimum fill level.

To evaluate the solutions produced by GAMMA-PC, a set of solutions was generated using the oldest, coldest (OC) strategy discussed in Sec. 1. These solutions were created using the *First Fit-T* packing heuristic and a schedule of loading dates.

Due to the unknown nature of the true loading plan, multiple schedules were used to generate the OC solutions. It was assumed that the dominant solution generated in the set would be similar to the oldest, coldest strategy used by industry.

GAMMA-PC performed well in relation to the OC solutions. Figures 4.1 and 4.2 presents the objective space of the Vermont Yankee problem and show the approximate set produced by GAMMA-PC after 166 generations in relation to solutions belonging to the OC set, the initial generation, and the 80th generation. Each dot represents the objective vector for one solution, and the histograms along the diagonal show the distribution of values for each objective function. The OC, 0, and 80 sets are each filtered to show only approximately 40% of the best solutions to avoid presenting too much information in the scatter plots. These plots show that GAMMA-PC began the search for nondominated solutions in the same area of the objective space as the OC solutions, which makes sense given that the *First Fit-t* packing strategy is one of the four main packing heuristics used in GAMMA-PC. The algorithm moved toward lower heat loads and lower transportation dates in set 80, and by the 166th generation, GAMMA-PC had found a diverse approximate set that dominates the oldest and coldest approach to dry cask loading.

Figure 4.1 also illustrates the relationship of the objective functions within F. The first and second objective functions have an indirect relationship that is almost linear. This makes sense given that the heat in each cask is a summation of the heat from the assemblies. The third objective does not exhibit variation. The maximum year to transport of 2039 is determined by the time required to meet the transportation decay heat limit of 0.272 kW per assembly, which cannot be improved regardless of the loading configuration. The OC solutions with f_3 in 2040 or later include casks with a concentration of higher burnup fuel and are determined by the time required to satisfy Eq. 2.18.

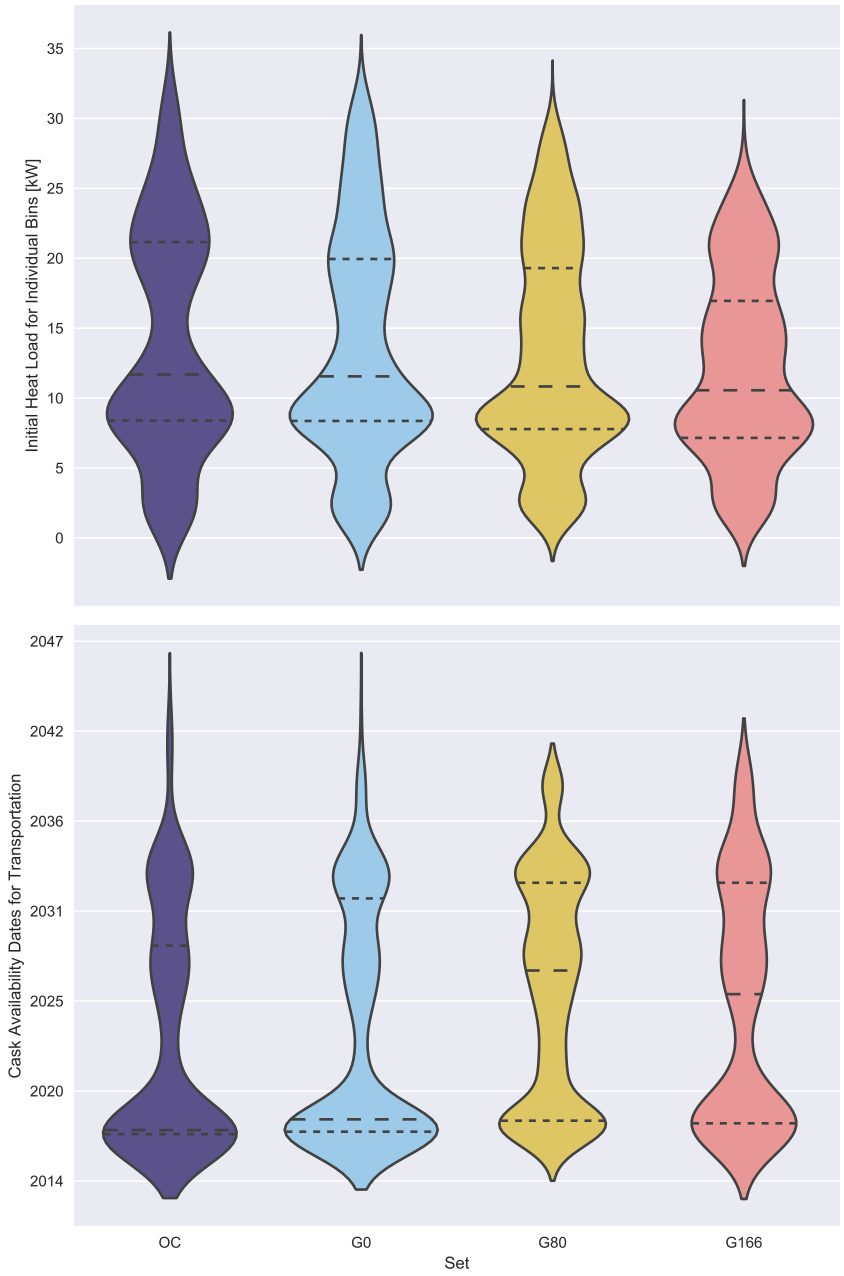


Figure 4.3: Violin plots comparing the cask characteristics of solutions for the Vermont Yankee dry cask loading problem. The OC solutions are compared to solutions found by GAMMA-PC during initialization, in generation 80, and for the approximate set (G166). The top plot shows the initial cask heat loads, and the bottom shows the time to transport for each cask.

Since the objective vector reduces the components of the dry cask loading problem to three simplified metrics, it is interesting to consider the details behind the vector. The first objective function is a straightforward sum of 0's and 1's, so the range shown in Figs. 4.1 and 4.2 gives a complete view of the number of casks in use. The other two functions collapse more data together.

Figure 4.3 presents that data in violin plots of cask initial heat loads and the time to transportation. A **violin plot** is similar to a box-and-whisker plot but uses a kernel density estimate distribution to form its sides instead of a box. The quartiles for each set in Fig. 4.3 are plotted on the violin, with the smaller dashed lines representing the 25% and 75% quartiles and the larger dashed line representing the median. Each violin represents the distribution of the characteristic for all of the casks in a solution, for all of the solutions within the given set. The top plot shows a reduction in the variability of the initial heat loads, although the median is not signficantly different from one distribution to another. The bottom plot shows that the median transport date of the approximate set is substantially later than for the OC set as a direct result of the reduction in the maximum.

Figure 4.4 filters the information in Fig. 4.3 to only show the dominant solution from the OC set ($F = [45, 13.70, 2041]$), the recommended GAMMA-PC solution, and its alternate. The recommended solution from the approximate set uses the minimum number of casks to anticipate the preferred plan for an ISFSI and is located at $F = [45, 13.48, 2039]$. The alternate solution includes one additional cask for every six months of transfer procedures for this scenario and is located at $F = [49, 12.42, 2039]$. The distributions of the initial heat load show similar medians for the sets, and the two GAMMA-PC solutions cover smaller ranges. The bottom plot shows the same trend as was present in Fig. 4.3, with the median time to transport much later for the GAMMA-PC solutions while the latest time is earlier. The alternate solution

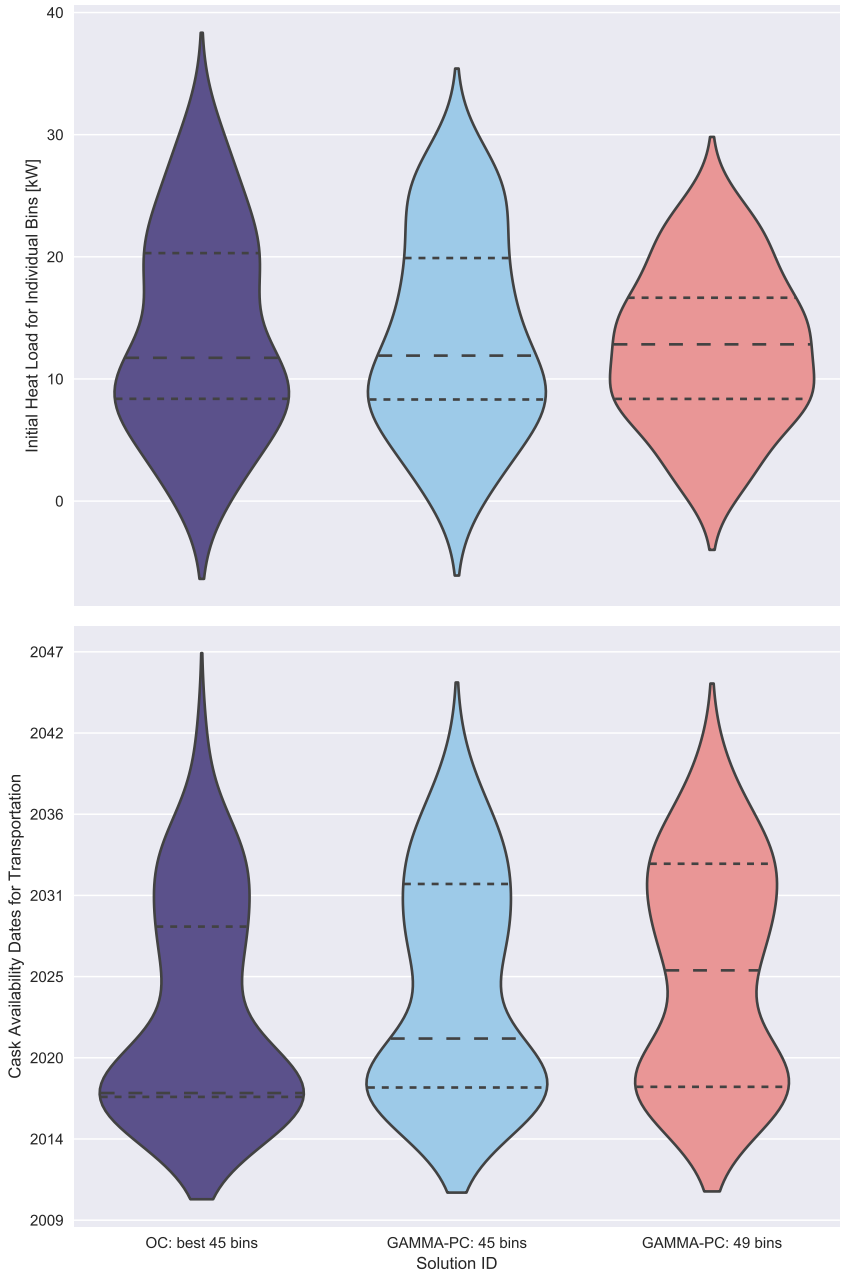


Figure 4.4: Violin plots comparing the cask-specific characteristics of the recommended Vermont Yankee loading plan and its alternate to an OC solution. The plot on top compares the initial heat load for each cask, and the plot on bottom compares the time at which each cask meets the transportation requirements.

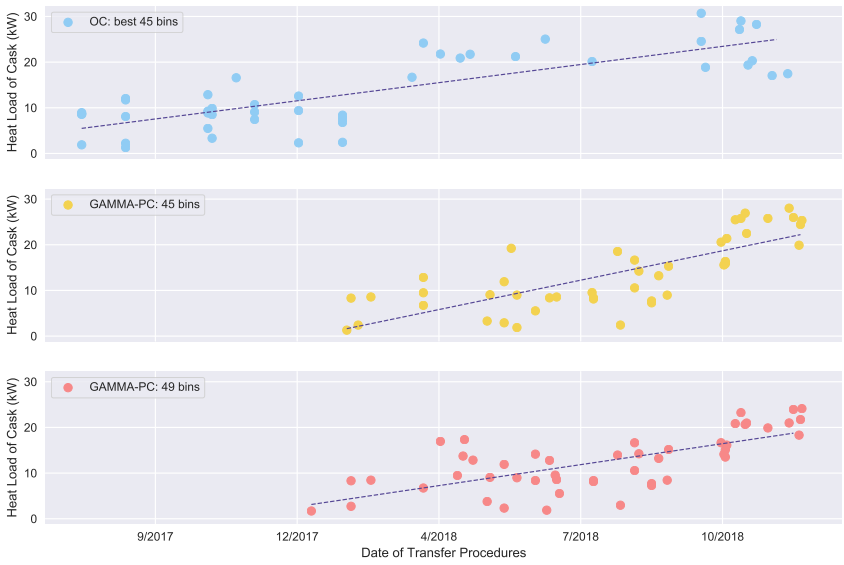


Figure 4.5: Scatter plots comparing the cask heat loads of the recommended Vermont Yankee loading plan and its alternate to an OC solution.

also has a later median compared to the recommended solution, although the 75% quartile is somewhat lower.

The top and bottom plots in Fig. 4.4 are replotted as functions of the $t_{fill,i}$ time in Figs. 4.5 and 4.7, respectively. All of the solutions plotted tend to fill hotter casks later, which is a reasonable strategy to allow for more nuclide decay in the most sensitive casks while loading those that are easier to handle. The trendlines for the GAMMA-PC solutions also end at lower heat levels than the OC trendline, mainly due to a reduction in the hottest casks. Figure 4.6 presents heat maps for the hottest cask in the OC solution and the recommended GAMMA-PC solution. The OC cask was loaded only two months before the GAMMA-PC cask, but its heat load is more than 2kW hotter. The GAMMA-PC cask features a more diverse mixture of assemblies and includes two open positions. Even though the OC cask holds more

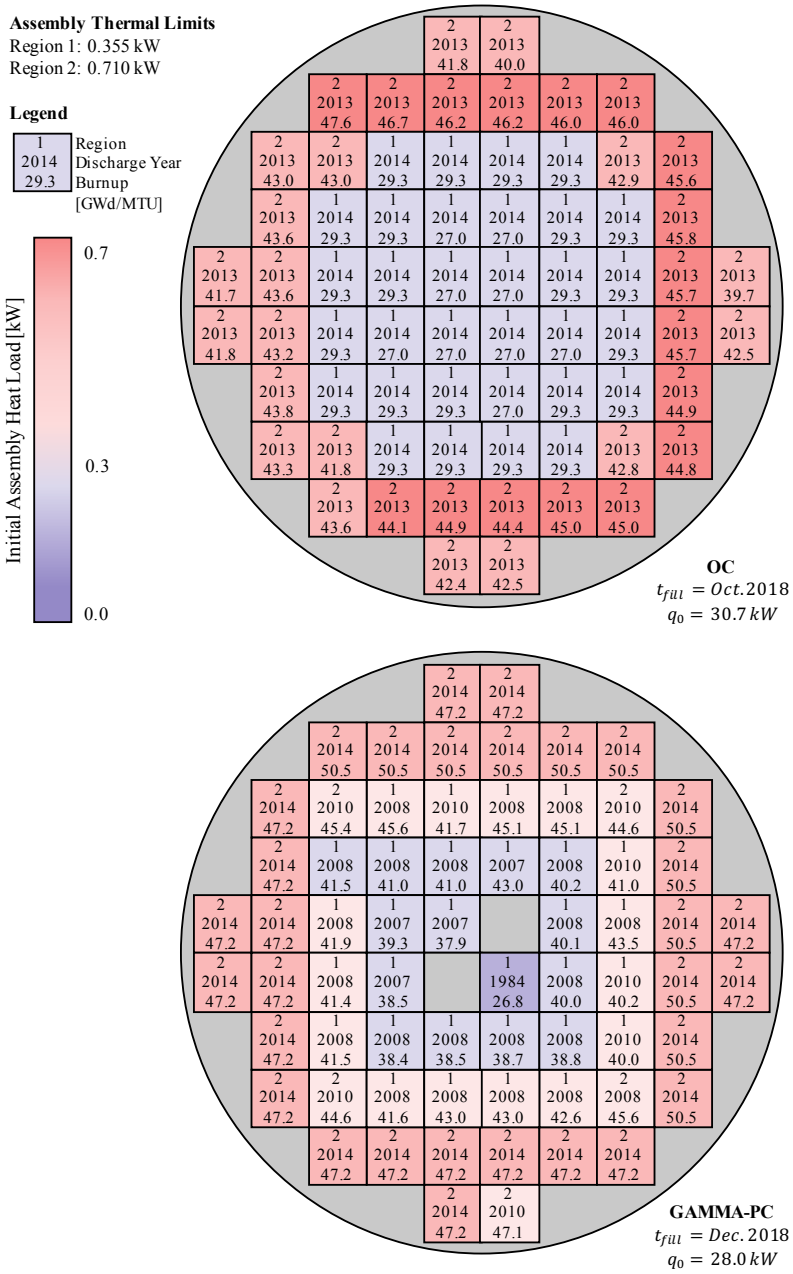


Figure 4.6: Heat maps for the hottest casks in the OC solution and in the recommended GAMMA-PC solution for the Vermont Yankee problem. Both casks are shown using the MPC-68 two-zone BWR preferential loading pattern. Note that the assemblies in this graph have only been optimized to the regional loading, and specific placements within the regions are arbitrary.

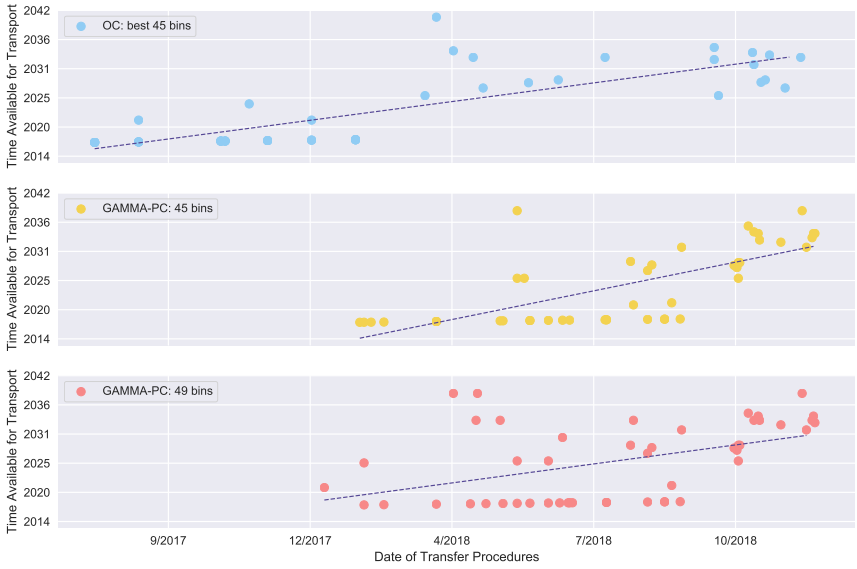


Figure 4.7: Scatter plots comparing the time to transport for the recommended Vermont Yankee loading plan and its alternate to an OC solution.

assemblies cooler than 0.3 kW, the GAMMA-PC cask achieves a much lower heat load overall. This suggests that when it is not possible to completely fill every cask, the empty positions should be strategically located. The additional casks used in the alternate solution lower the hottest heat loads even more.

Figure 4.7 compares the transportation eligibility dates for the OC, the recommended, and the alternate solutions. Many of the casks in all three solutions are eligible as soon as they are loaded, which makes sense given the amount of older fuel present at Vermont Yankee. Almost all of the eligibility dates after 2020 are determined by the cooling time required for individual assemblies to reach 0.272 kW. It is impossible to find a configuration with a lower minimum date than 2039 in this scenario, although it is possible to reduce the number of casks that become eligible in those years. However, concentrating hot assemblies into fewer casks has a

detrimental effect on their initial heat load.

4.2 Comanche Peak

The Comanche Peak Nuclear Power Plant poses a more complex packing problem than Vermont Yankee. The site operates two PWRs that came online in 1990 and 1993 and have operating licenses that expire in 2030 and 2033 [150,151]. The reactors share a spent fuel pool, and UNF-ST&DARDS includes fuel projections for the pair until 2053. It was assumed in this scenario that Comanche Peak would receive a license extension to correspond with the fuel projections. Unlike Vermont Yankee, the Comanche Peak spent fuel pool faces capacity constraints along the optimization timeline, requiring a number of casks to be filled before certain dates. Altogether, the Comanche Peak dry cask loading problem represents the most complete example in this research of the constraints developed in Sec. 2 and demonstrates how GAMMA-PC performs when applied to a long-term planning mission.

To better understand the effect of the constraints, the Comanche Peak problem was optimized using two different "bin" types, described in Table 4.3. The first type was the cask system currently used in the Comanche Peak ISFSI, the HI-STORM 100 MPC-32. This cask type holds 32 assemblies, and 227 casks would be required to fully store all of the current and projected used fuel. The second cask type was the HI-STORM FW MPC-37. This cask system would maintain the current utility-vendor relationship and allow for a discussion of how the loading plan might change based on a higher capacity cask. The minimum number of MPC-37 to hold the assemblies would be about 30 fewer than in the MPC-32 scenario. The HI-STAR 190 transportation package for the MPC-37 has not received the final approval from the NRC yet, so the parameters shown in Table 4.3 were taken from the submitted Safety Analysis Report [152].

Table 4.3: Dry system characteristics for the Comanche Peak Nuclear Power Plant cases.

	Comanche Peak	
Cask System	HI-STORM 100 MPC- 32	HI-STORM FW MPC-37
C_{cask}	32	37
$H_{s,max}$ [kW]	36.7	45.0
$H_{t,max}$ [kW]	20.0	31.82
$BU_{s,max}$ [GWd/MTU]	68.2	68.2
$BU_{t,max}$ [GWd/MTU]	31.2	68.2
No. of Casks Needed:		
Lower Bound	227	196
Maximum $\overline{M}$	302	261
Function Evaluations	7500	7500

This section compares the GAMMA-PC results for each scenario with OC solutions and then compares them to each other. Because of the greater restrictions imposed on the problem by the constraints, significantly more computational time was devoted by GAMMA-PC in these scenarios to finding *packable* bins and feasible $t_{fill,i}$ values compared to either the Vermont Yankee or Zion problems. Therefore, the results presented here were achieved after the completion of the lowest number of function evaluations to ensure a reasonable runtime.

4.2.1 Using the HI-STORM MPC-32 Cask System

GAMMA-PC performed well compared to the OC solutions for the Comanche Peak loading problem. Figures 4.8 and 4.9 show the approximate set produced after 46 generations in relation to the top 10 fronts from the initial generation (0), the 20th generation, and the OC solutions. Each dot represents an objective vector, and the histograms along the diagonal of the scatter matrix show the distribution of values within the four sets. As with Vermont Yankee, the initial generation began in the same region of the objective space as the OC solutions. The intermediate generation shows more progress in achieving lower heat loads and broader diversity,

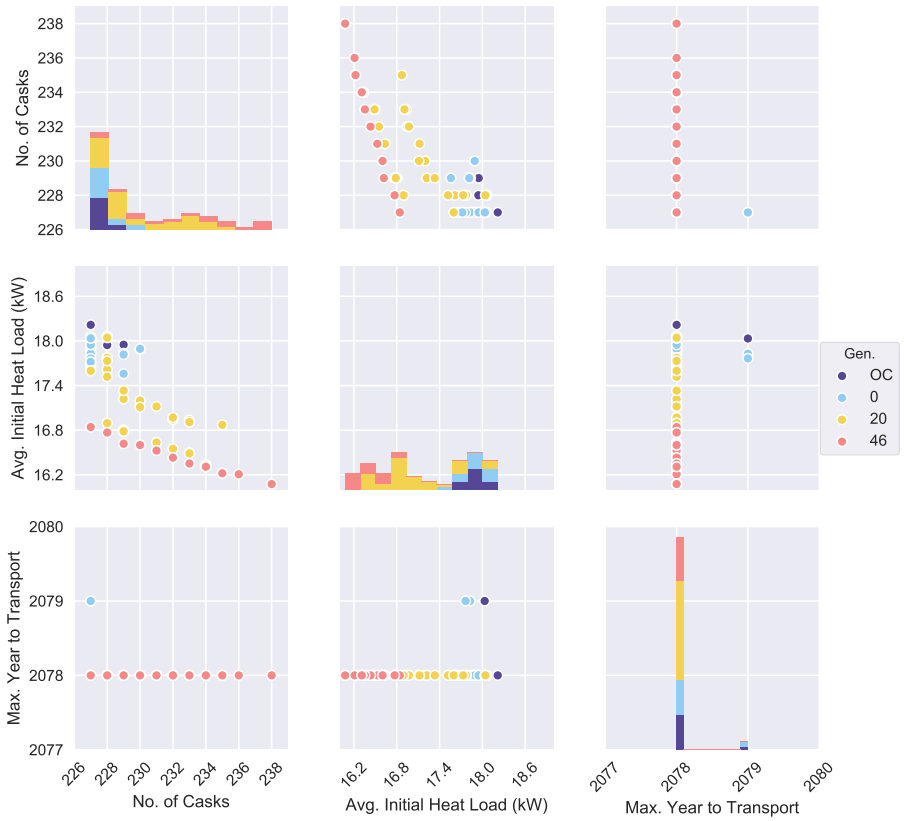


Figure 4.8: Scatter matrix showing objective vectors for GAMMA-PC and for OC solutions in the Comanche Peak loading problem (MPC-32). This figure compares the approximate set found by GAMMA-PC in generation 46 with the initial generation (Gen. 0), an intermediate generation (Gen. 20), and OC solutions.

and the packing solutions in the approximate set are substantially cooler than the OC solutions.

The scatter matrix in Fig. 4.8 also illustrates the relationship between the objectives for the Comanche Peak problem. As expected, the initial heat load decreases as the number of casks used increases. The latest year to become eligible for transportation also shows a constant value for all of the solutions except a few in the initial generation and the OC set. This date in 2078 is determined by the cooling time required for some assemblies discharged in 2053 to reach the 0.625 kW per as-

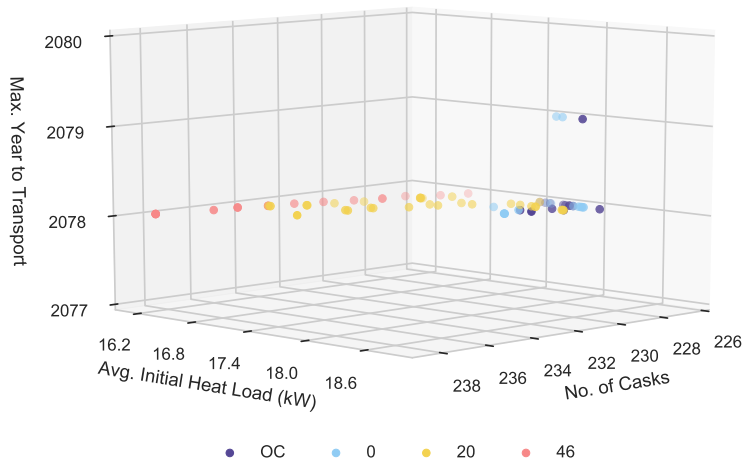


Figure 4.9: Scatter plot showing objective vectors for GAMMA-PC and for OC solutions in the Comanche Peak loading problem (MPC-32). This figure compares the approximate set found by GAMMA-PC in generation 46 with the initial generation (Gen. 0), an intermediate generation (Gen. 20), and OC solutions.

sembly transportation decay heat limit. As with Vermont Yankee, this value cannot be improved by the loading configuration with the MPC-32.

Figure 4.10 presents violin plots of the individual cask distributions in the solution sets. The top plot shows that while the initial distribution of heat loads found by GAMMA-PC covered a wider range than the OC solutions, the search narrowed to solutions with lower maximum and lower medians heat loads. The distributions of the time to transport also show a decrease in the variability for the GAMMA-PC solutions. This indicates that as the algorithm searched for a way to lower the value of f_3, the earliest time to transport for GAMMA-PC was pushed back while the median date was moved somewhat earlier in time. However, this increase did result in both later minimums and medians compared to the OC distribution.

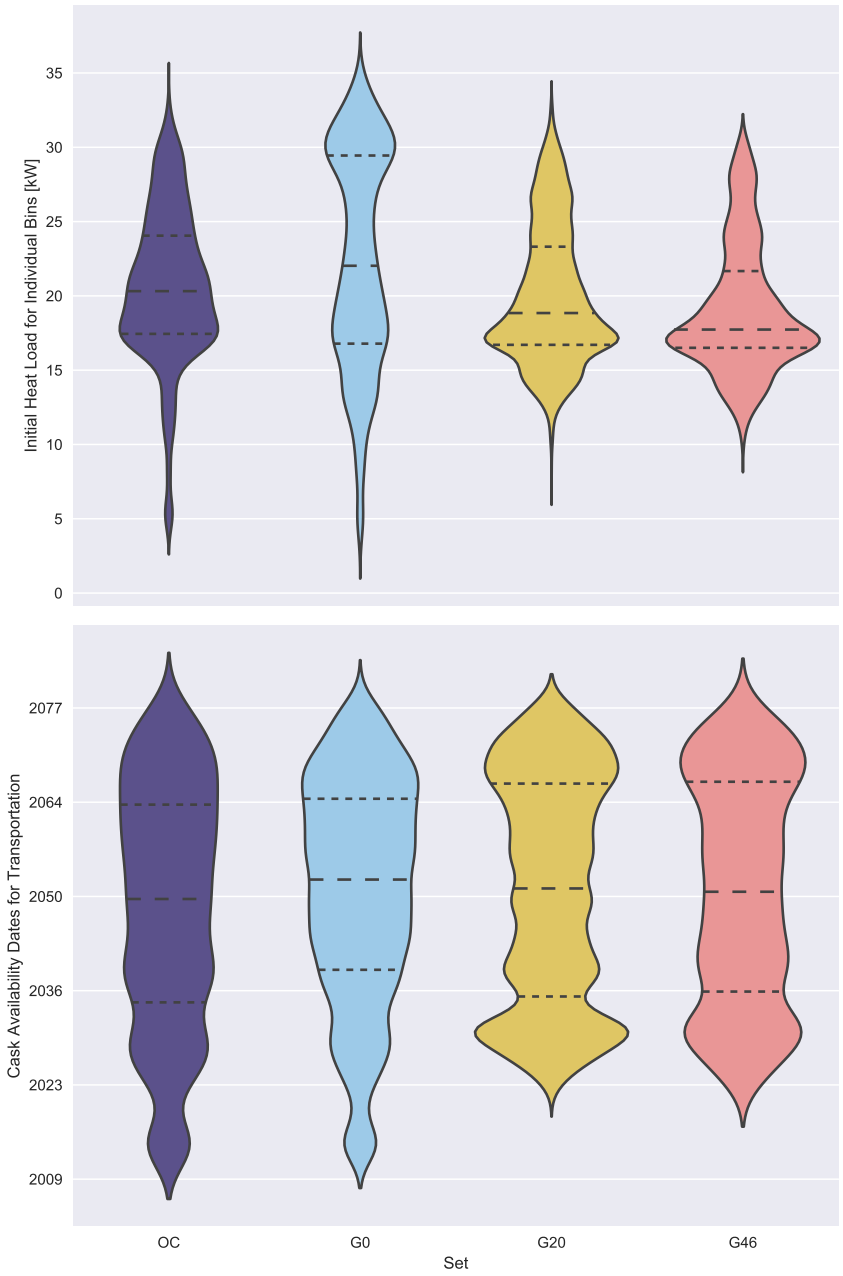


Figure 4.10: Violin plots comparing the cask characteristics of solutions for the Comanche Peak dry cask loading problem (MPC-32). The OC solutions are compared to solutions found by GAMMA-PC during initialization, in generation 20, and in the approximate set (G46). The top plot shows the distributions of initial cask heat loads, and the bottoms shows the eligibility dates for transportation.

Figure 4.11: Violin plots comparing the cask-specific characteristics of the recommended Comanche Peak loading plan (MPC-32) to the dominant OC solution. The plot on the left compares the initial heat load for each cask, and the plot on the right compares the time at which each cask meets the transportation requirements.

The information in Fig. 4.10 is filtered in Fig. 4.11 to only show the dominant solution from the OC set ($F = [227, 17.60, 2078]$) and the recommended solution from GAMMA-PC ($F = [227, 16.84, 2078]$). The recommended solution is the configuration in the approximate set with the minimum number of casks in use. The median initial cask heat load of the recommended solution is significantly cooler than for the OC solution. While the coolest cask is much hotter, the top 50% hottest casks are cooler by at least 2 kW. The width of the violins in the top quarter are also wider than at the very bottom, indicating that more casks are cooler in the hottest segment than are hotter in the coolest. On the other hand, the distributions of the cask transportation eligibility dates share many of the same features, with equivalent maximums and virtually equivalent medians.

The distributions of heat load and $t_{trans,min,i}$ values are replotted as functions of

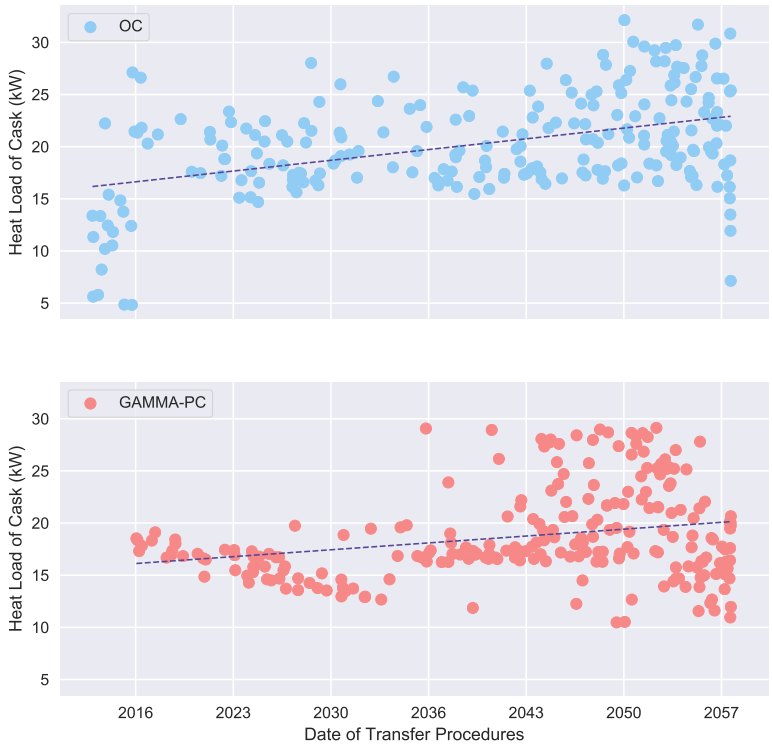


Figure 4.12: Scatter plots comparing the time-dependent initial cask heat loads of an OC solution with the recommended solution for Comanche Peak (MPC-32).

the $t_{fill,i}$ dates in Figs. 4.12 and 4.14. Figure 4.12 shows that the hottest casks are loaded later, as discussed in the Vermont Yankee scenario, although the trend in this scenario is not purely increasing. This change from Vermont Yankee makes sense given the pool capacity issue and that only the last 5 years of the optimization timeline allow for partially-filled casks, whereas the entire Vermont Yankee timeline was within the decommissioning window. The trendline for the recommended GAMMA-PC solution is shown to end approximately 3 kW lower than the trendline for the OC solution, partly due to the decrease in the heat of the hottest casks. Additionally,

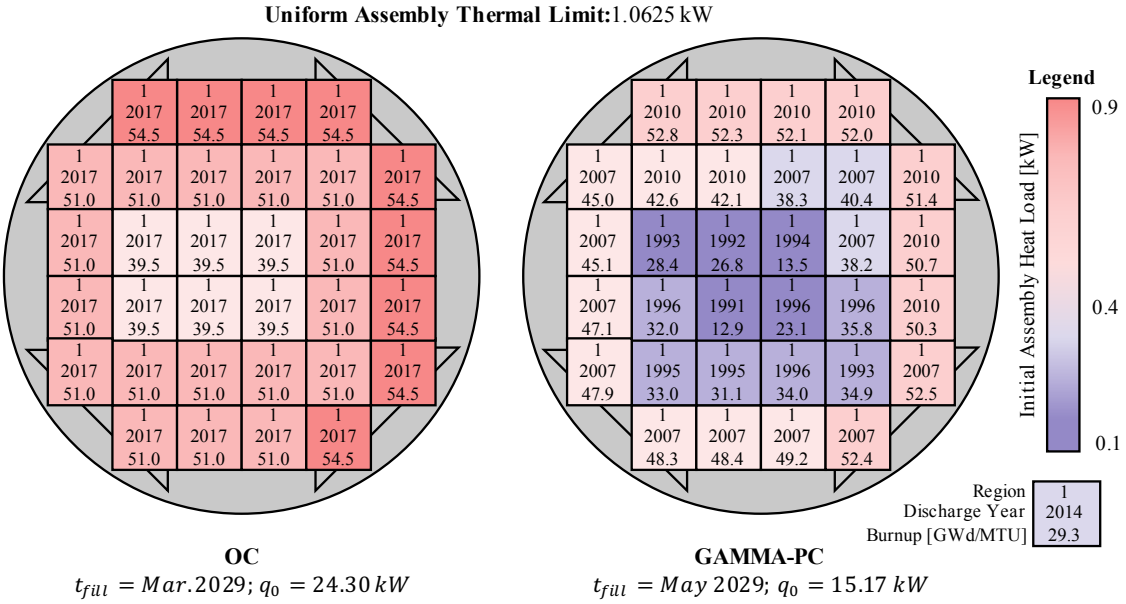


Figure 4.13: Heat maps for the hottest casks loaded in 2029 in the OC solution and in the recommended solution for Comanche Peak (MPC-32). Both casks are shown using the uniform PWR loading pattern. Assemblies in this map were optimized to the regional loading, and specific placements within the regions are arbitrary.

the casks filled before 2036 were much cooler than the OC casks loaded in the same timeframe, although an increase in the spread of values after 2036 is observed, which is due to a need to remove assemblies from the pool.

Figure 4.12 presents a prime example of the question posed in Sec. 1: how might a more sensible mixture of used fuel assemblies impact the long-term aspects of dry storage? The OC plot shows that a number of casks were filled at the beginning of the timeline with heat loads lower than 15 kW, which suggests the immediate short-term benefit to workers of much lower dose levels. However, the removal of those cold assemblies from the pool resulted in uniformly hotter casks filled in the long term.

This is further illustrated in Fig. 4.13 by considering the assembly-level heat in

Figure 4.14: Scatter plots comparing the transportation eligibility dates of an OC solution with the recommended GAMMA-PC solution for the Comanche Peak loading problem (MPC-32).

the hottest cask loaded in 2029, the year before the older reactor's current license expiration. The hottest cask filled that year in the OC solution is much hotter than the hottest cask for the recommended solution. While the GAMMA-PC cask does not include assemblies as recently discharged as the OC cask, which gives the GAMMA-PC an unequal advantage in this comparison, if it were filled with only assemblies that were discharged in 2010, it could be up to 4 kW hotter. The diversity of the assemblies in the GAMMA-PC cask help reduce its overall heat load, while the OC

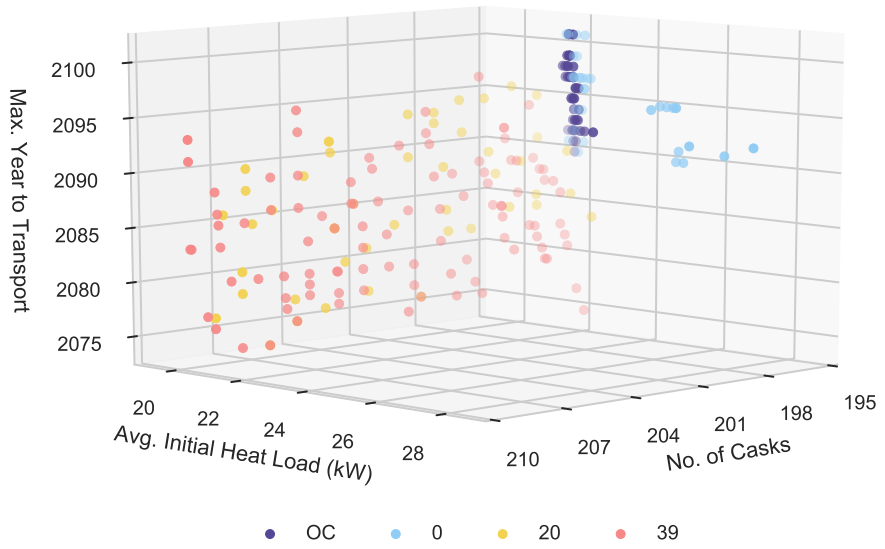


Figure 4.15: Scatter plot showing objective vectors for GAMMA-PC and for OC solutions in the Comanche Peak loading problem (MPC-37). This figure compares the approximate set found by GAMMA-PC in generation 39 with the initial generation (Gen. 0), an intermediate generation (Gen. 20), and OC solutions.

cask is filled purely with the casks from the same discharge cycle.

Figure 4.14 compares the transportation elgibility dates for the OC and recommended solutions. The trends are similar, and the distributions are not significantly different. This coincides with the finding for the latest dates–this characteristic is mainly assembly-determined rather than dependent on the loading configuration.

4.2.2 Using the HI-STORM MPC-37 Cask System

The Comanche Peak loading problem has a much more complex objective space landscape when the MPC-37 is used to store the used fuel. Figures 4.15 and 4.16 illustrate the objective vectors for this problem. Some of the same trends under the MPC-32 scenario are shown here. GAMMA-PC started its search for solutions in approximately the same area of the objective space as the OC solutions and

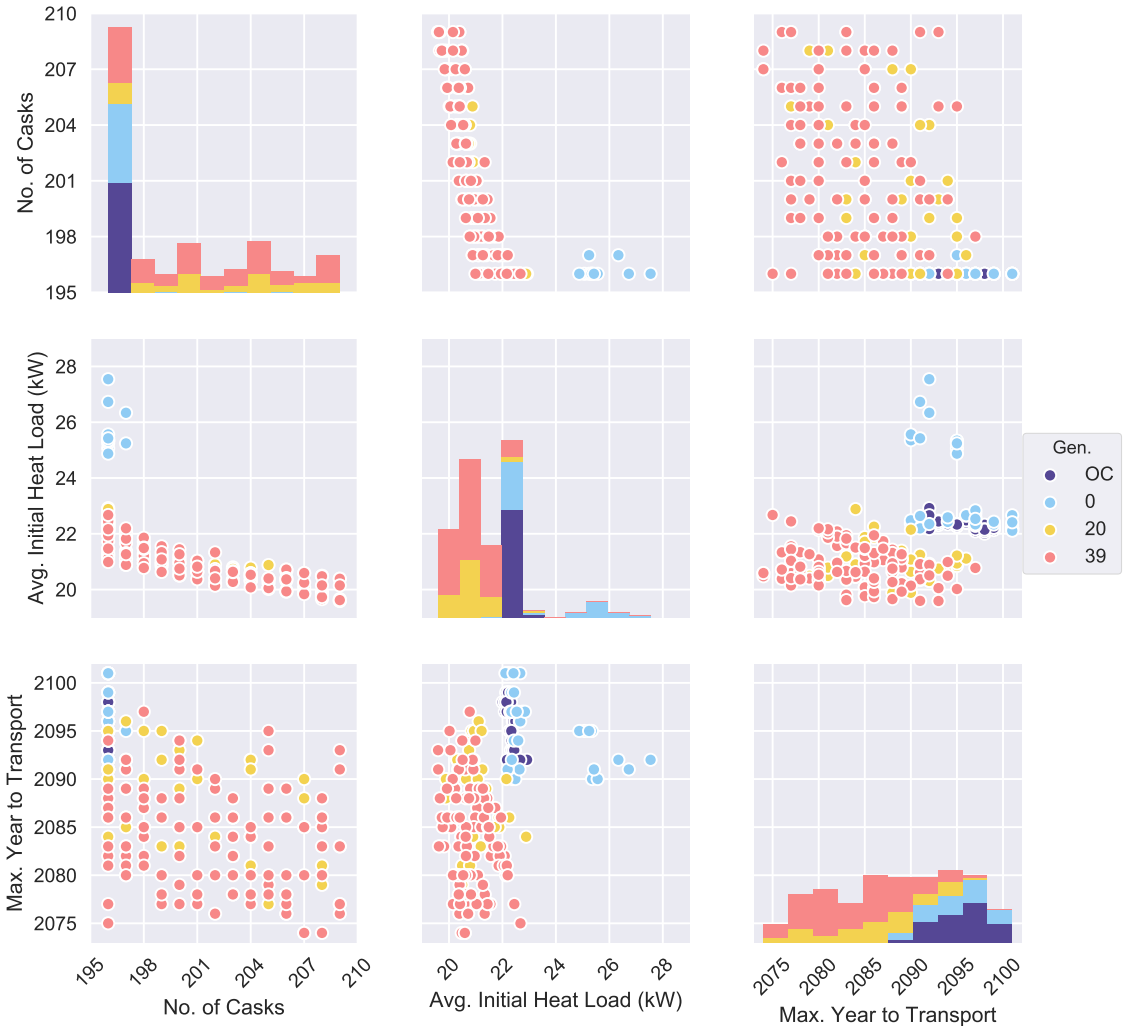


Figure 4.16: Scatter matrix showing objective vectors for GAMMA-PC and for OC solutions in the Comanche Peak loading problem (MPC-37). This figure compares the approximate set found by GAMMA-PC in generation 39 with the initial generation (Gen. 0), an intermediate generation (Gen. 20), and OC solutions.

moved toward areas with lower weighted average initial heat loads with more diversity along the x-axis. Notable differences include a shift upward in the heat load and a substantial increase in the variability of f_3. The increase in heat is expected given the higher capacity of the MPC-37. The time to transport is more complicated.

The broad variability of f_3 shown in Fig. 4.16 exhibits a slight pattern when viewed in the 3-dimensional space of Fig. 4.15. The cloud of objective vectors may look random, but the vectors in the lower area of the x-axis show some solutions grouped together along invisible curves.

Based on this evidence, Figure 4.17 was created to investigate this area of the objective space as planar slices along the x-axis from 196 to 199 casks in use. The plots in this figure show a strong indirect relationship between the second and third objective functions. Unlike Vermont Yankee and Comanche Peak (MPC-32), the third objective value in this scenario is primarily determined by Eq. 2.18, meaning that the time to transport is dependent on the loading configuration. While it may seem counterintuitive that solutions with lower average initial heat loads would have longer cooling periods before becoming eligible for transport, it is a reasonable relationship considering the length of time between $t_{fill,i}$ and $t_{trans,min,i}$. With burnup and mass held constant, the hotter an assembly is the higher the activity and thereby the higher the rate of nuclide decay. Therefore, the assembly would have fewer unstable nuclides after twenty years of cooling and a lower level of dose, while the initially cooler assemblies would have more remaining. While the assemblies from Comanche Peak do not have uniform burnup levels, this effect is seen in the hottest and latest casks to be ready for transport.

While GAMMA-PC performed much better at finding solutions with lower objective vectors than the OC solutions in this scenario, the individual cask characteristics behind the vectors do not show huge differences. Figure 4.18 presents the distribu-

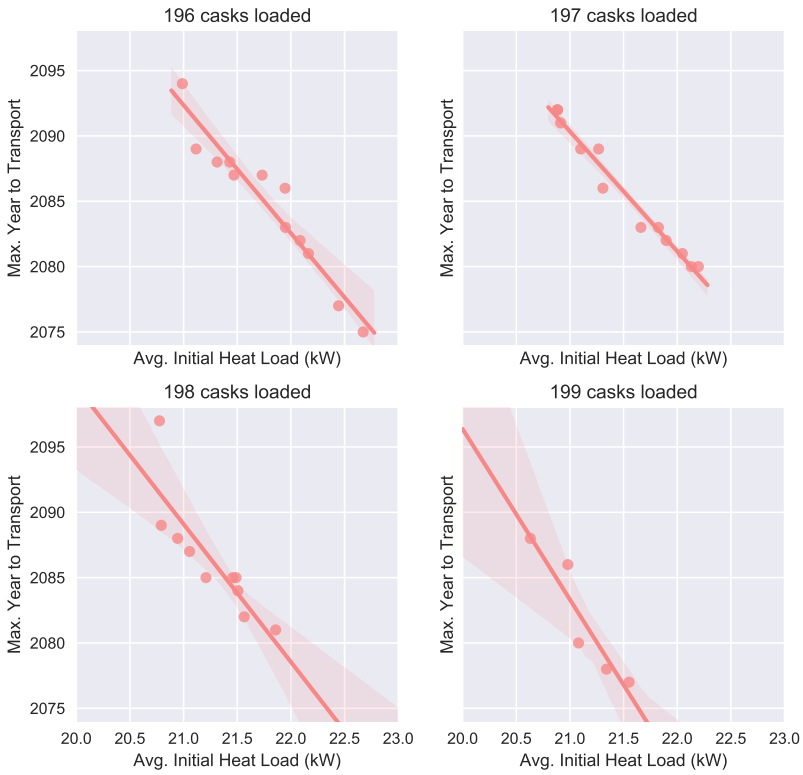


Figure 4.17: Scatter plots showing objective vectors for solutions from the GAMMA-PC approximate set split out by f_1 in the Comanche Peak loading problem (MPC-37). Lines of regression show the indirect relationship of f_2 and f_3, and the translucent bands show the 95%confidence interval.

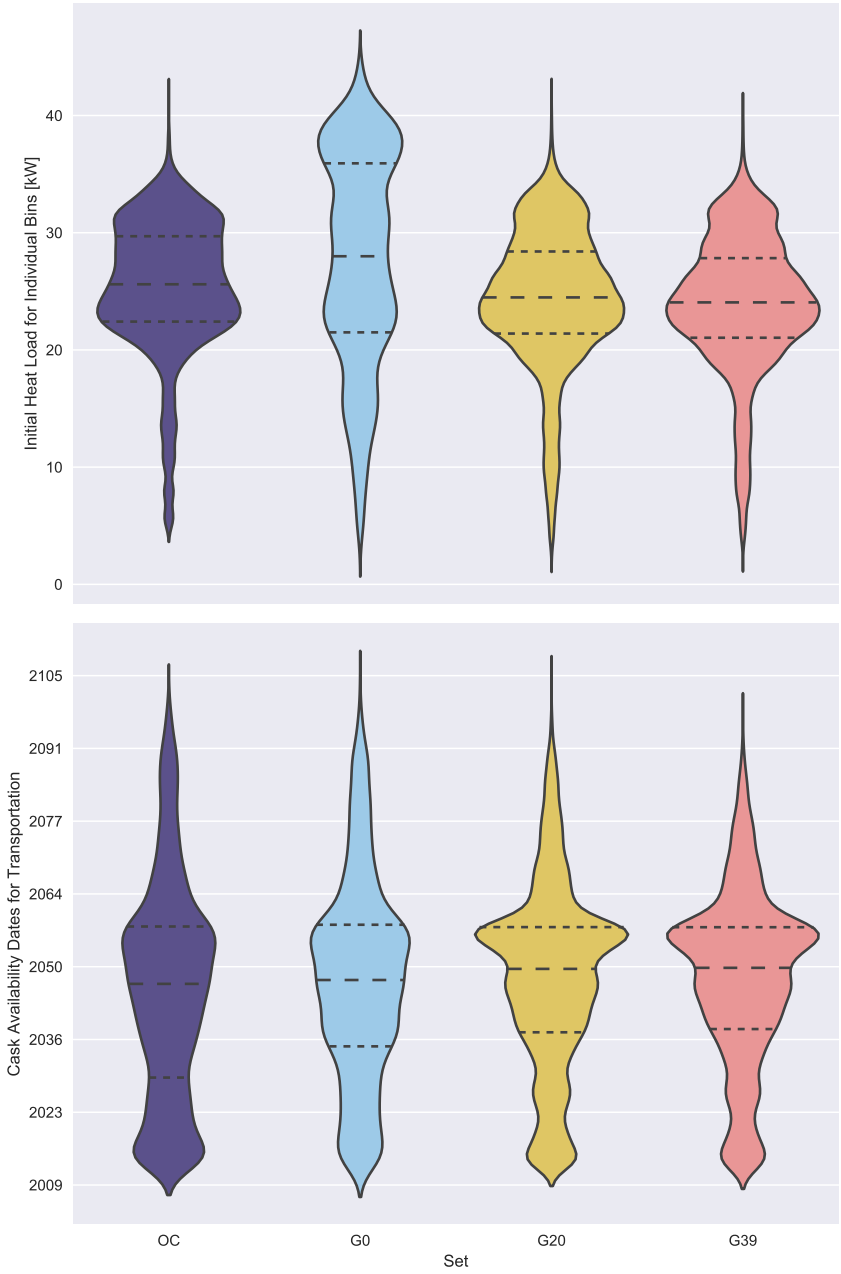


Figure 4.18: Violin plots comparing the cask characteristics of solutions for the Comanche Peak dry cask loading problem (MPC-37). The OC solutions are compared to solutions found by GAMMA-PC during initialization, in generation 20, and in the approximate set (G39). The top plot shows the distributions of initial cask heat loads, and the bottoms shows the eligibility dates for transportation.

tions of the cask heat loads and the eligibility dates for transportation. The distribution of the cask heat load for the initial generation starts off much broader than the distribution for the OC set. However, the distribution of the intermediate generation covers approximately the same range with a similar shape, and the approximate set shows very few differences. The same pattern is seen in the distributions of $t_{trans,min}$. The maximum for the approximate set is slightly earlier than for the OC set and the median is slightly later. Overall, the plot shows that while GAMMA-PC achieved better fitness values over time, the approximate set is not remarkably different than the OC set in this regard.

Figure 4.19 filters this information to only show the dominant OC solution ($F = [196, 22.0, 2098]$), the recommended solution, and its alternate from the approximate set. In this case, the recommended and the alternate solutions both employ the minimum number of casks, and the difference lies between their f_2 and f_3 values. At a weighted average of 21.0 kW, the recommended solution has the lowest initial heat of those with $f_1 = 196$ but the latest time to transport (2094). The alternate solution has the earliest time to transport (2075) but the hottest average initial cask heat load at 22.7 kW. This is reflected in Fig. 4.19. The distributions for the recommended solution are almost identical to the OC solution. The median heat load is slightly cooler, and the 25% quartile in the time to transport is slightly later than the OC statistics, but overall the difference is negligible. The alternate solution on the other hand shows a wider range of initial heat loads with a much smaller range of eligible dates. The median values for both of its distributions are subtly higher, but the latest date to transport is about 20 years earlier.

These distributions are replotted as functions of $t_{fill,i}$ in Figs. 4.20 and 4.22. Figure 4.20 shows the distributions of the initial heat load with trendlines overlaid. The scatter plot for the recommended solution is similar to the OC solution, with a

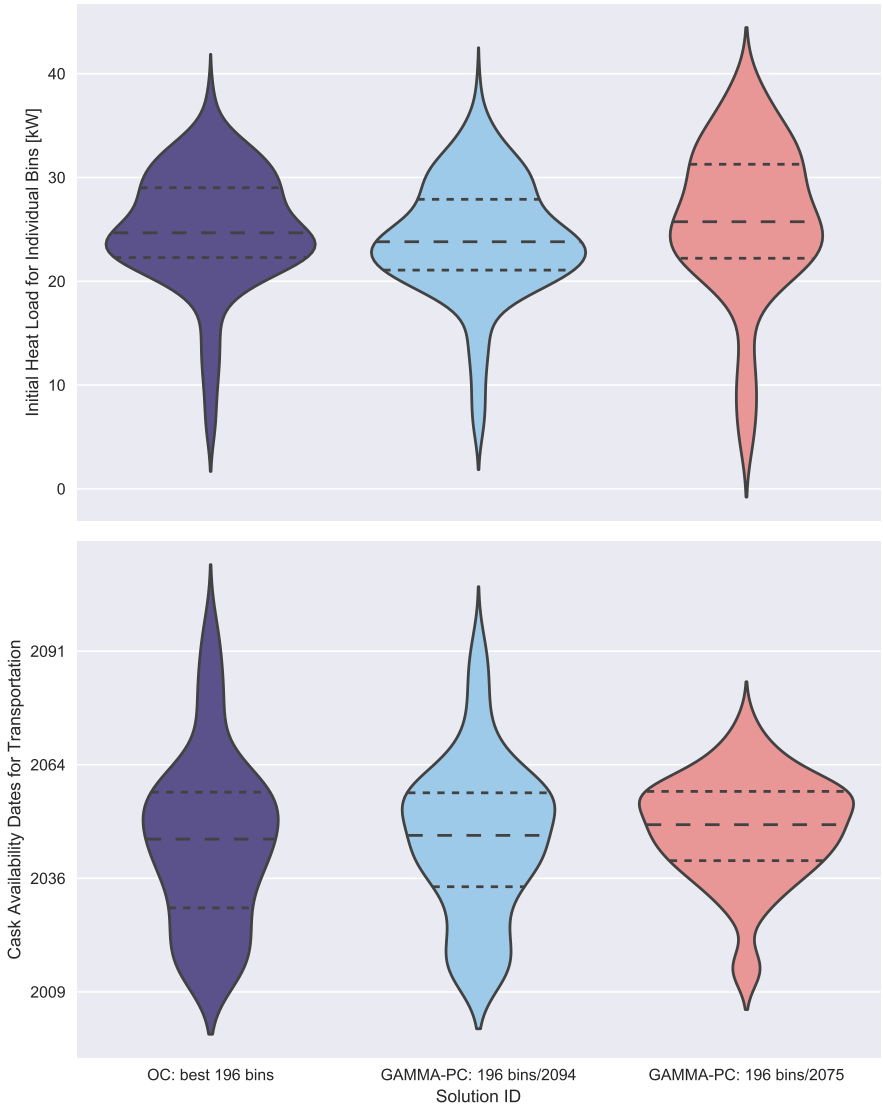


Figure 4.19: Violin plots comparing the cask-specific characteristics of the recommended Comanche Peak loading plan (MPC-37) and its alternate to an OC solution. The plot on top compares the initial heat load for each cask, and the plot on bottom compares the time at which each cask meets the transportation requirements.

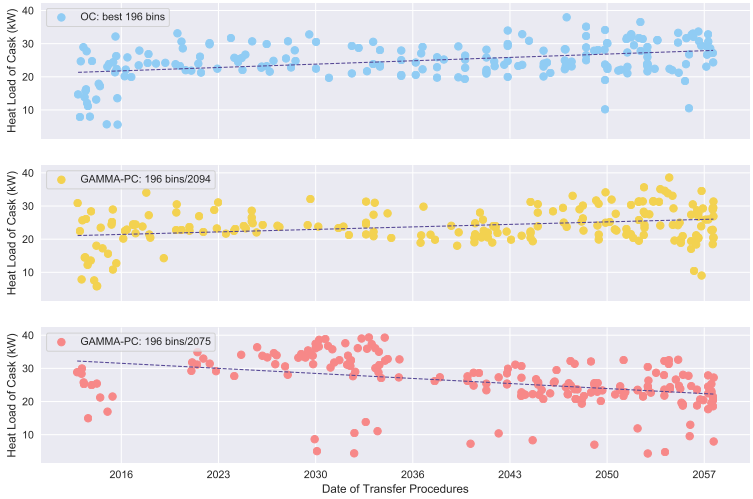


Figure 4.20: Scatter plots comparing the time-dependent initial cask heat loads of an OC solution with the recommended GAMMA-PC solution for the Comanche Peak loading problem (MPC-37).

slight reduction in the trendline in 2057 due to an increased presence of cooler casks. Figure 4.21 illustrates this similarity with the hottest casks packed during 2029 for each solution. The GAMMA-PC cask is slightly cooler than the OC cask but shows the same level of uniformity in assembly selection. The scatter plot in Fig. 4.22 for the alternate solution is substantially different than the others, showing a decreasing trendline with the hottest casks filled during the 2023 to 2036 timeframe.

Figure 4.22 shows the distributions of the transportation eligibility dates as a function of $t_{fill,i}$ for the three solutions. The scatter plots of the OC and recommended solutions again show similar patterns, although the latest date in the recommended loading is 4 years earlier than the latest date in the OC solution. Unlike the other two plots, the scatter plot for the alternate solution shows that the casks with the latest transportation dates are within the 2023 to 2036 timeframe, which

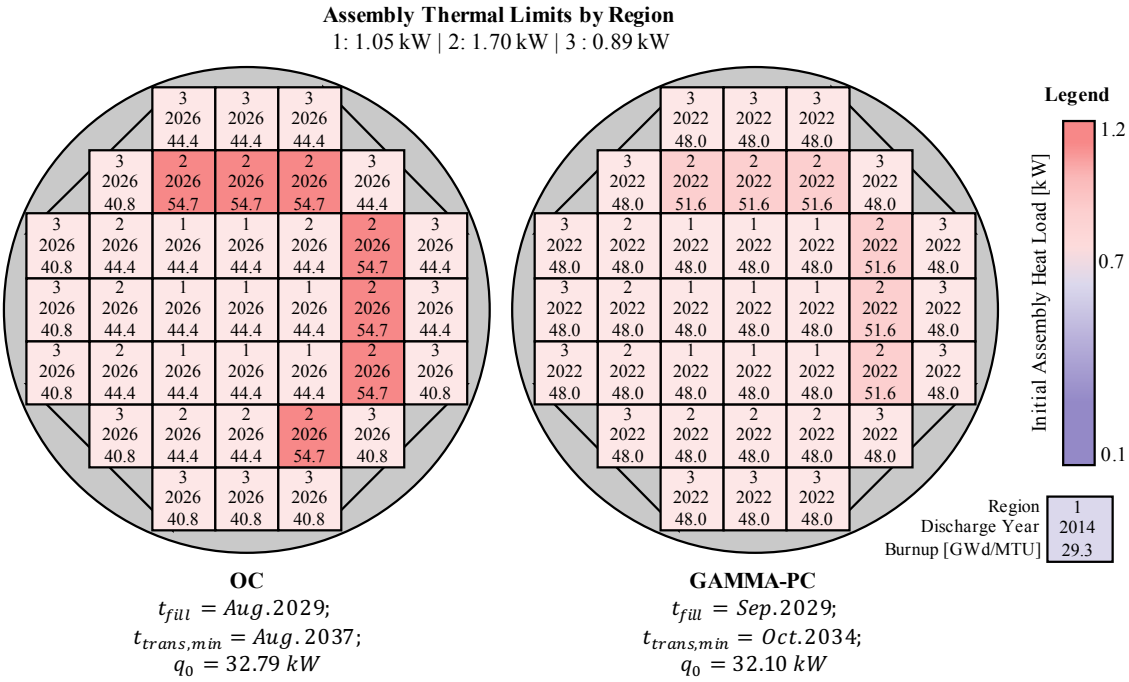


Figure 4.21: Heat maps for the hottest casks loaded in 2029 in the OC solution and in the recommended GAMMA-PC solution for the Comanche Peak problem (MPC-37). Both casks are shown using the MPC-37 three-zone PWR preferential loading pattern. Note that the assemblies in this graph have only been optimized to the regional loading, and specific placements within the regions are arbitrary.

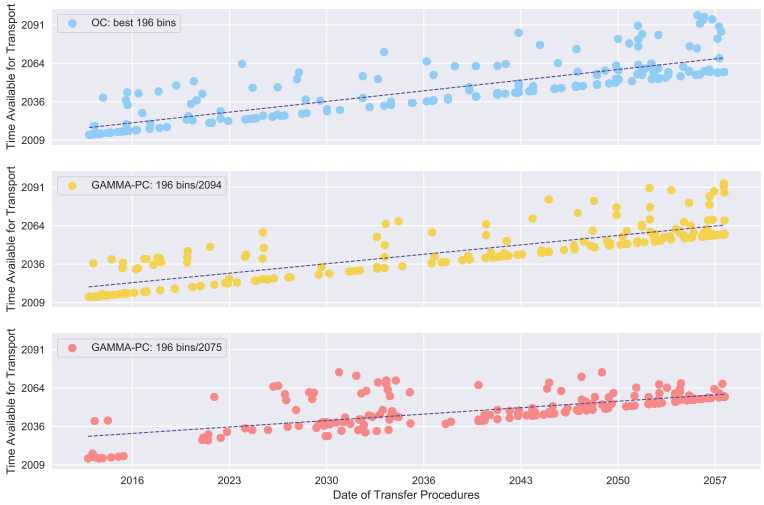


Figure 4.22: Scatter plots comparing the transportation eligibility dates of an OC solution with the recommended GAMMA-PC solution for the Comanche Peak loading problem (MPC-37).

makes sense given that the casks loaded during this timeframe would be the most active casks and would need more time to cool. Despite the longer cooling times up front, these casks are ready for transport approximately two decades earlier than the latest casks in the other two solutions.

4.2.3 Comparing Results for MPC-32 and MPC-37

The general relationships between the objective functions shown thus far can be extended to the choice between the MPC-32 cask and the MPC-37 cask for the Comanche Peak dry cask loading problem. The MPC-32 cask is the chosen system by the Comanche Peak Nuclear Power Plant, and GAMMA-PC was able to find loading configurations with it that significantly outperformed the traditional oldest and coldest strategy. The second optimization of Comanche Peak with MPC-37 was performed to investigate if it would be possible to enable the faster removal of

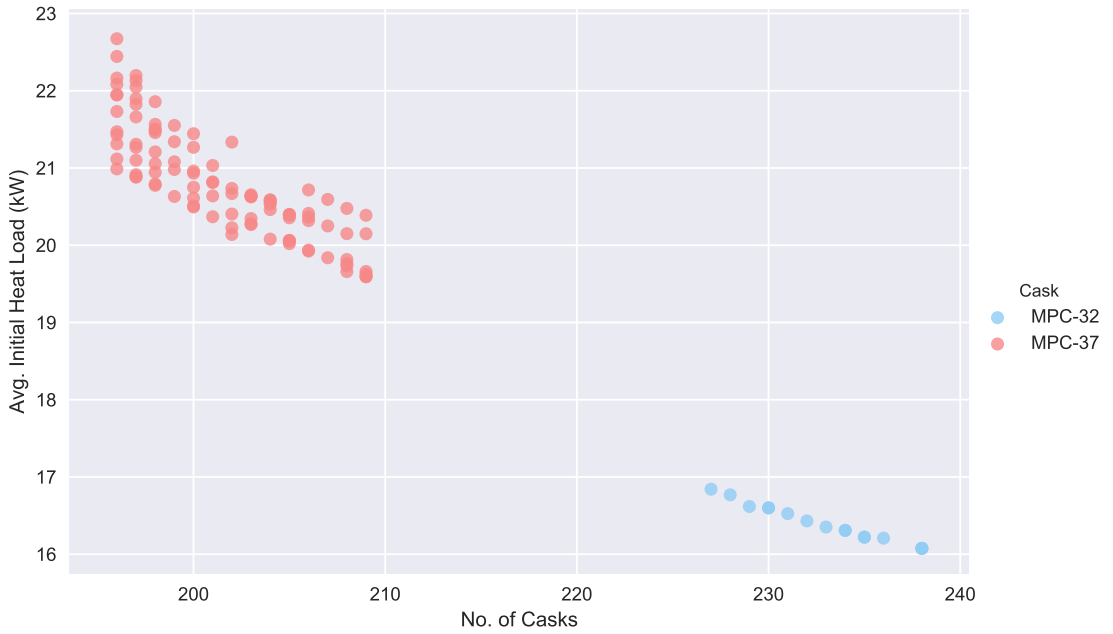


Figure 4.23: Comparison of the relationship between the number of casks and the average initial heat load for the approximate sets found by GAMMA-PC for Comanche Peak using the HI-STORM MPC-32 and MPC-37 cask systems.

used fuel from the ISFSI assuming the existence of centralized storage location. The results show that it would be possible given certain tradeoffs.

Figure 4.23 presents a comparison of the approximate sets found by GAMMA-PC for both scenarios when only considering the relationship between the first and second objective functions. This figure maintains the indirect relationship between the number of casks and the initial heat load and even suggests that the two sets might belong to the same trend if a regression curve for either were extrapolated. The increase in average initial heat load for the MPC-37 is expected with the increase in capacity, and this scatter plot illustrates the straightforward trade off between the two systems. A smaller number of casks would be easier to move but would also result in higher levels of heat and thereby higher activity during transfer procedures.

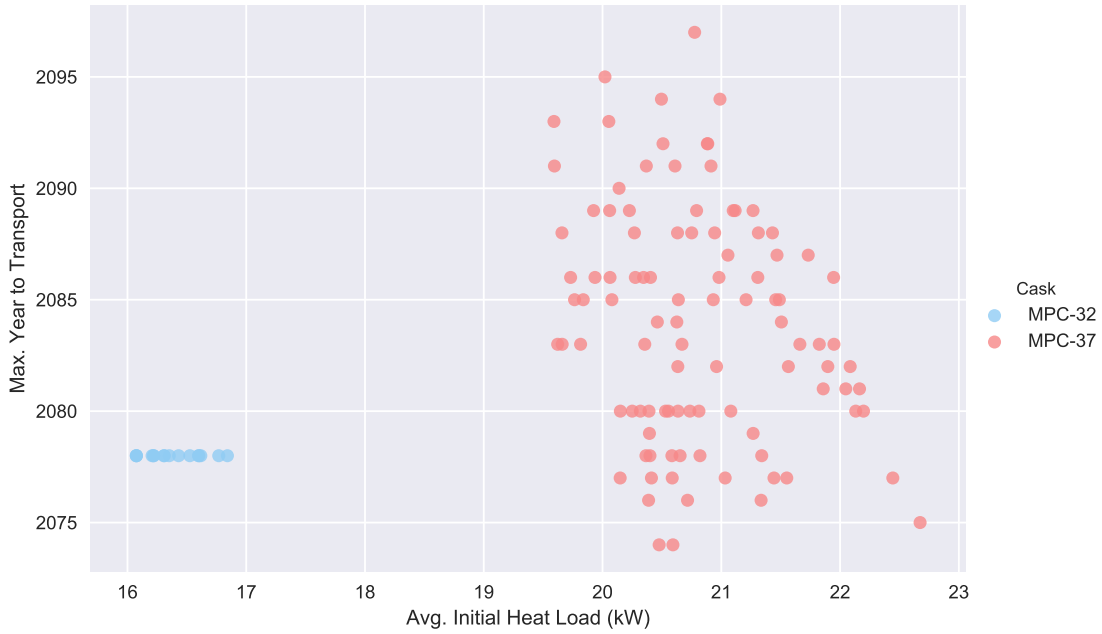


Figure 4.24: Comparison of the relationship between the average initial heat load and the maximum year to transport for the approximate sets found by GAMMA-PC for Comanche Peak using the HI-STORM MPC-32 and MPC-37 cask systems.

Figure 4.24 shows the comparison through the relationship of the second and third objective functions. The trade offs are more subtle here. For the MPC-32, there is no correlation between the heat of the cask and the latest date of eligibility in 2078. The approximate set for the MPC-37, on the other hand, shows the possibility of an earlier latest date if the configuration were carefully chosen. A number of the MPC-37 solutions fall before 2078, but the majority are ready later than that. The hottest solution in the group ready before 2078 is the alternate solution at $f = [196, 22.7kW, 2075]$, and the coldest is a solution at $f = [209, 20.1kW, 2077]$. The tradeoff presented in this graph is that an earlier last transportation date is possible with the MPC-37 in return for much higher heat loads, which can be mitigated somewhat through the use of additional casks. If the utility were to consider the

switch in cask types, it would need to decide if the tradeoff would be worth it to remove the fuel one to four years faster.

4.3 Zion

The Zion dry cask loading problem presents a unique opportunity to test GAMMA-PC. Zion Nuclear Power Station entered decommissioning in 1998 and began transferring used fuel assemblies to dry storage in December 2013 [123]. As discussed in Sec. 1.3, the transfer project optimized the load plan to reduce the dose at the site boundary, which required an iterative set of calculations between CASKLOADER, storage dose classifications, and transport cask criticality assessments. The transfer procedures concluded in January 2015, making it possible to evaluate GAMMA-PC against a real world solution of the loading problem.

The optimization of the Zion assembly selection was performed twice with GAMMA-PC. The first application of the algorithm maintained the original transfer timeline of approximately one year and used the modified version of the F_p constraint listed in Table 4.1 to more closely approximate the real transfer frequency. The second run extended the transfer timeline to a 10 year horizon to examine the effect of a more gradual loading strategy. Both scenarios used the NAC MAGNASTOR-

Table 4.4: Dry system characteristics for the Zion Nuclear Power Station.

	Zion
Cask System	NAC MAGNASTOR
C_{cask}	37
$H_{s,max}$ [kW]	35.5
$H_{t,max}$ [kW]	23.0
$BU_{s,max}$ [GWd/MTU]	60.0
$BU_{t,max}$ [GWd/MTU]	70.0
No. of Casks Needed:	
Lower Bound	61
Maximum $\overline{M}$	81
Function Evaluations	20,000

MAGNATRAN system chosen by ZionSolutions as the cask model [123]. Table 4.4 lists its characteristics. This system uses an inner transportable storage canister that can be placed within a storage overpack or a transfer cask [153]. The MAGNATRAN transfer cask is currently under review, so the transport characteristics in Table 4.4 were copied from the submitted CoC [154].

The GAMMA-PC results for both scenarios are compared to the real loading strategy, which will be referred to as the "basis" in this discussion. This solution stores the fuel within 61 MAGNASTOR canisters, and the $t_{fill,i}$ value for each canister is the date it was loaded. Since the canisters can hold 37 PWR used fuel assemblies, the 61 containers used by ZionSolutions represent a loading configuration with the least amount of open space possible [123]. Inserting the basis into the objective vector F results in an average initial heat load slightly lower than 14.0 kW, and the last cask would be ready to be transported in 2025.

4.3.1 Using the Original Timeline

GAMMA-PC performed well against the basis solution under the original transfer timeline. Figure 4.25 shows the comparison in a two dimensional scatter matrix of the objective space. Each dot represents the objective vector for one solution, and the histograms along the diagonal show the distribution of values for each objective function. Two sets of solutions from GAMMA-PC are plotted to show the evolution of its exploration. The first is the first five fronts of the initial population of solutions produced by GAMMA-PC, and the second is the approximate set saved after GAMMA-PC reached its ending criteria in generation 178. The basis solution can only be seen in the histograms due to overlapping with the GAMMA-PC results. It is dominated by the GAMMA-PC results, although the distance to the nearest solution in the approximate set ($F = [61, 13.84, 2025]$) is remarkably close given that

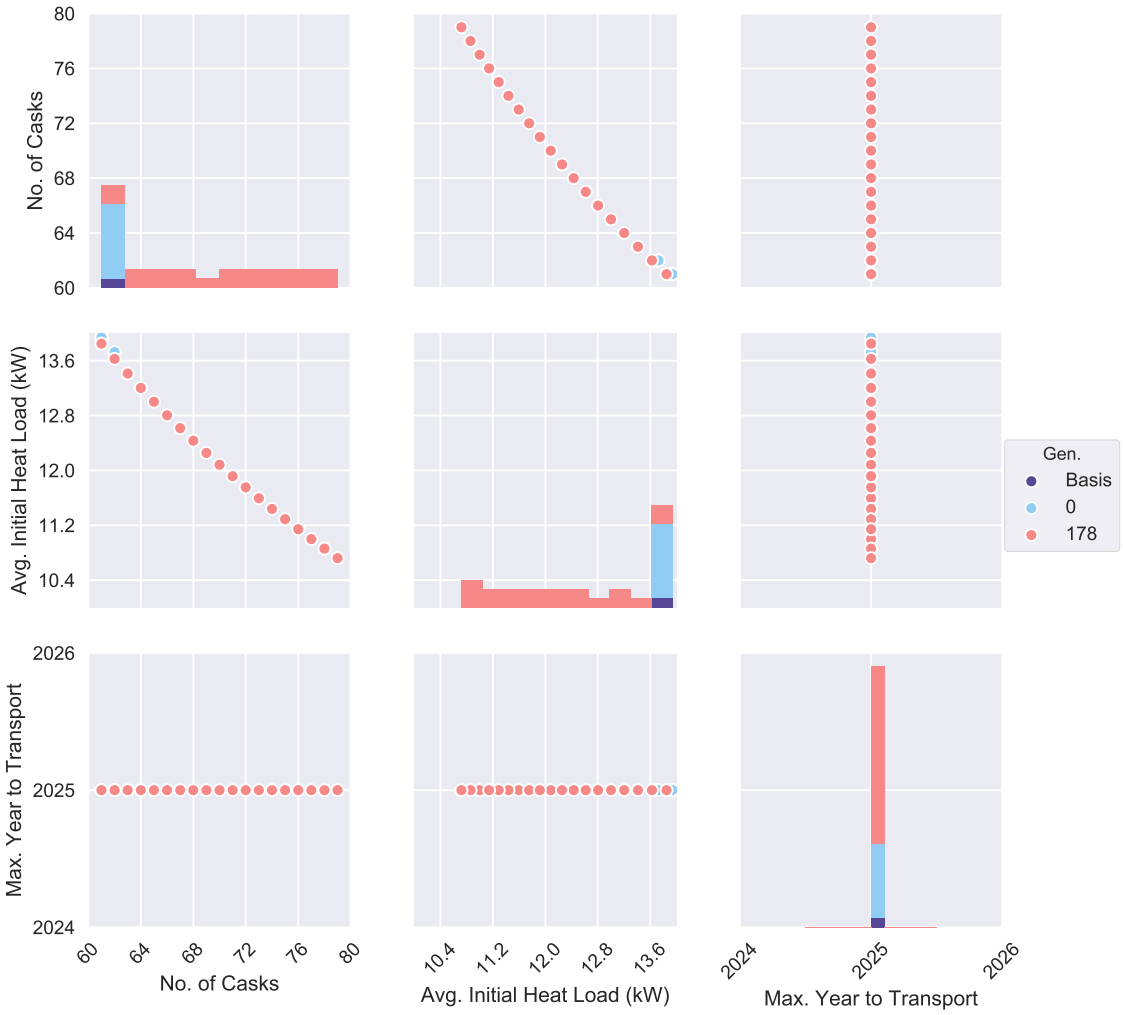


Figure 4.25: Scatter matrix of the objective space for the Zion dry cask loading problem using the original loading timeline. This plot shows the evolution of the solutions found by GAMMA-PC from the initial generation to the approximate set, found in generation 178. The basis solution is located at $[61, 13.94, 2025]$.

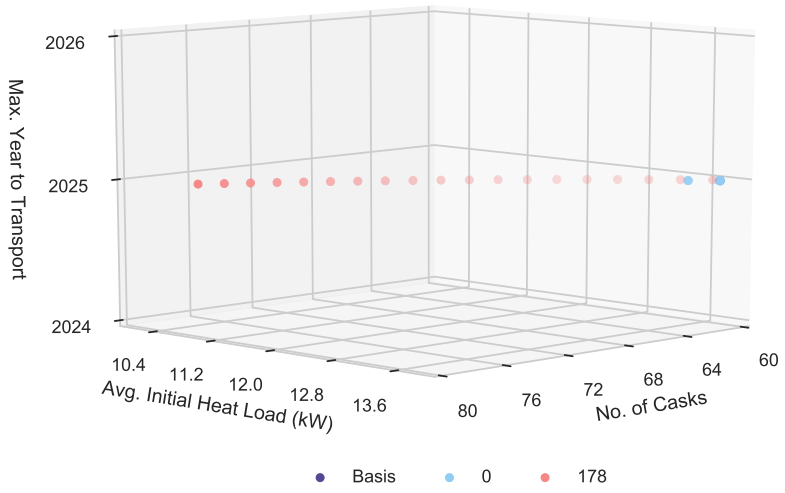


Figure 4.26: Scatter plot of the objective space for the Zion dry cask loading problem using the original loading timeline. This plot shows the evolution of the solutions found by GAMMA-PC from the initial generation to the approximate set, found in generation 178. The basis solution is located at $[61, 13.94, 2025]$.

the basis solution was optimized using a different objective.

Figure 4.25 also illustrates the relationship of the objective functions within F. The first and second objective functions have an indirect relationship that is almost linear. This makes sense given that the heat in each cask is a summation of the heat from the assemblies. The third objective shows that the Zion loading problem has one possible value and cannot be optimized to find a lower maximum time to transport. The year 2025 is determined by an assembly-specific cooling time requirement from the transportation loading curve and cannot be changed regardless of the loading configuration. In this way, the Zion loading scenario reduces to a bi-objective optimization problem. Figure 4.26 reiterates this, showing the approximate

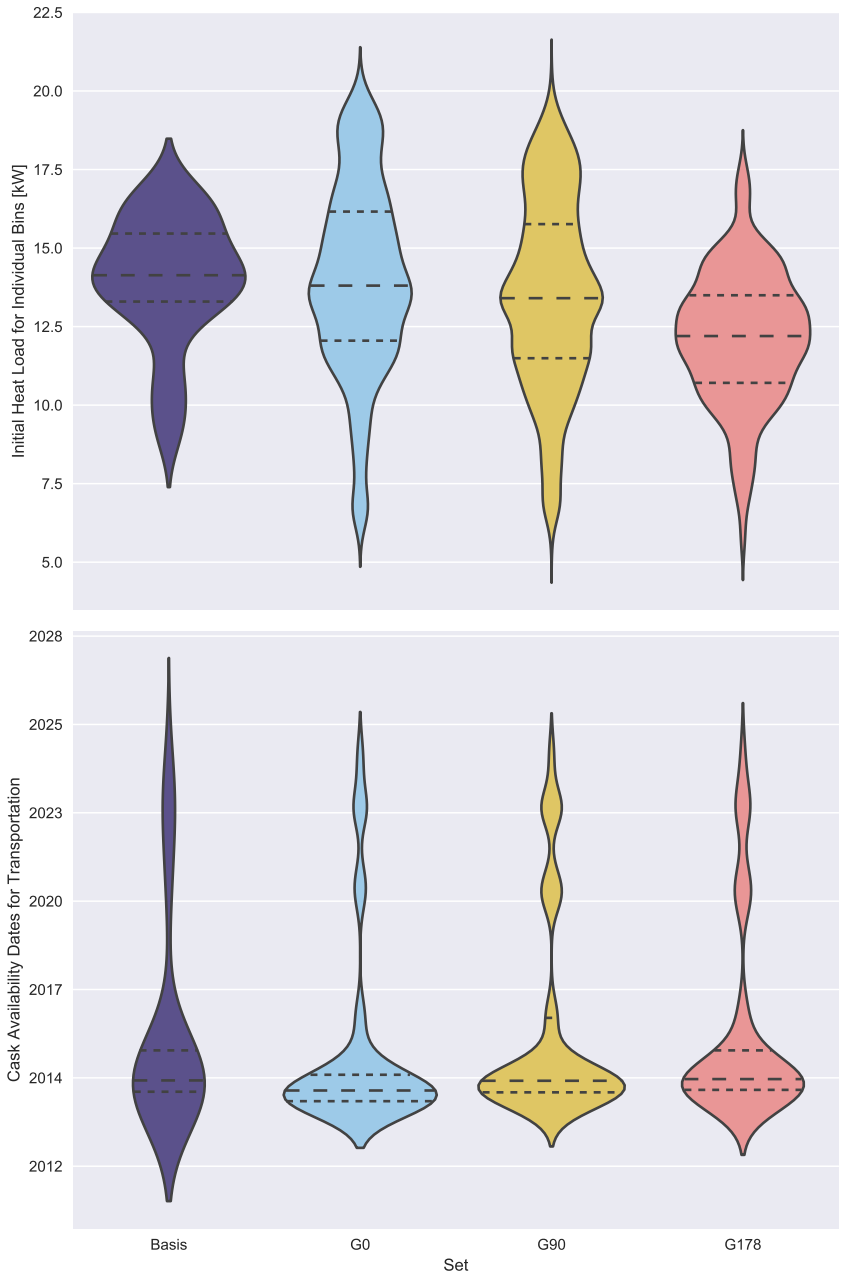


Figure 4.27: Violin plots comparing the cask characteristics of solutions for the Zion dry cask loading problem under the original timeline. The basis solution is compared to solutions found by GAMMA-PC during initialization, in generation 90, and in the approximate set (G178). The top plot shows the distribution of the initial heat loads, and the bottom shows the eligibility dates for transportation.

set as a curve within the 2025 plane of objective three.

Figure 4.27 presents violin plots of cask initial heat loads and time to transportation. The GAMMA-PC violins show that the initial population started off with a wide range of initial heat load values, and the algorithm gradually moved toward areas of the feasible region with lower heat loads. The cask heat loads in the approximate set are significantly lower than the basis solution, which makes sense given that the distribution includes solutions with more than 70 casks in use. The distribution of the time to transport is relatively unchanged through the progression of the algorithm, and the violin of the basis and the approximate sets are similar.

Figure 4.28 filters the information shown in Fig. 4.27 to only include the recommended and alternate loading plans from GAMMA-PC. The recommended loading plan is the solution from the approximate set that uses the minimum number of casks. This solution most closely corresponds to the setup of the basis. The alternate loading plan is a solution from the approximate set that uses the minimum number of casks plus one cask for every quarter of the transfer procedures. The initial heat load distributions show that the recommended solution covers a wider range and has a somewhat smaller median. While the top 50% hottest casks in the recommended solution are hotter than the same set for the basis, the bottom 50% are much cooler. The distribution for the alternate plan is lower overall. The distributions for the time to transport are almost identical.

Figures 4.29 and 4.30 show the cask characteristics as a function of the $t_{fill,i}$ dates. The former plot focuses on the initial heat load of the casks, and the latter on the eligible date for transport. Both plots include trendlines. In Fig. 4.29, the trendlines for the basis and the recommended solution are similar if slightly shifted in time, and there is more variability in the solution found by GAMMA-PC. The trendline for the alternate solution is lower overall. The shift in heat loads for the

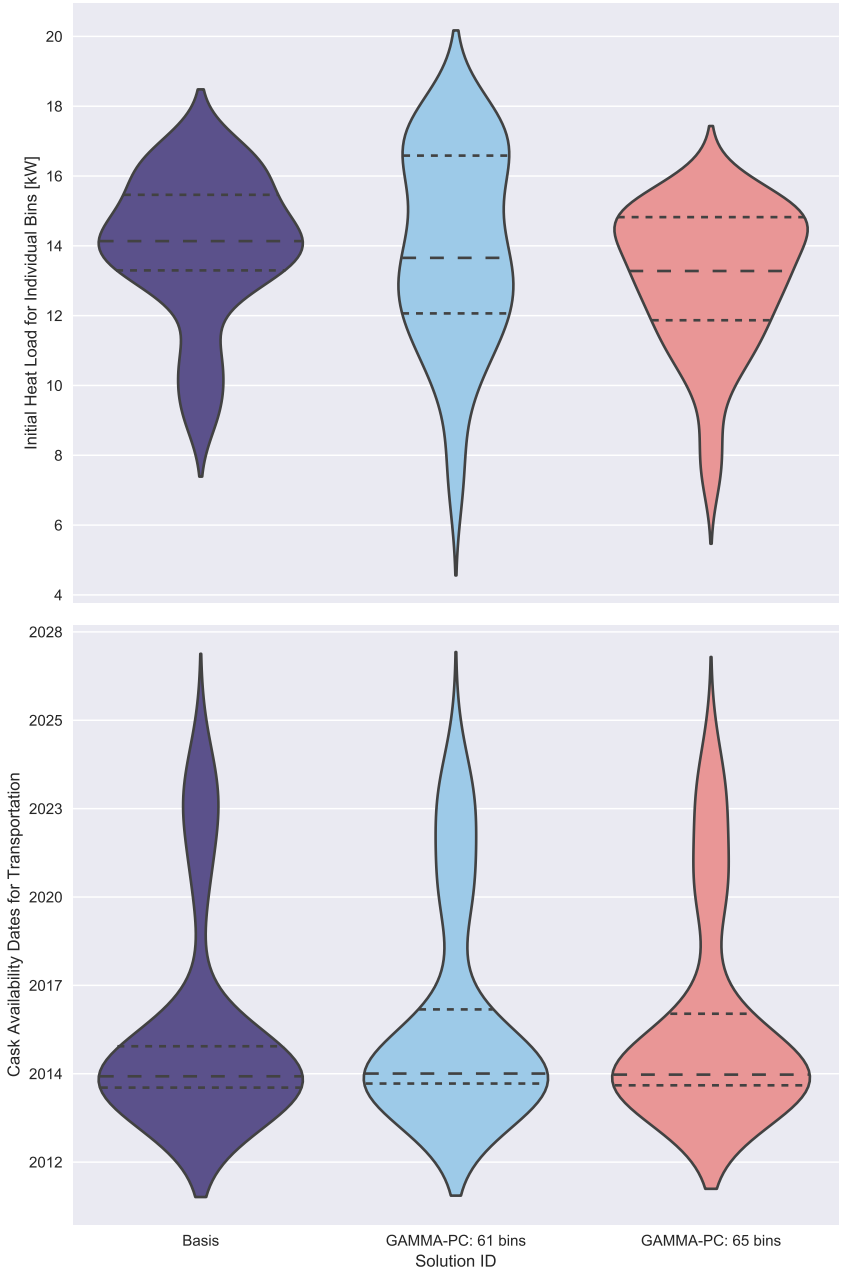


Figure 4.28: Violin plots comparing the cask-specific characteristics of the recommended and alternate Zion loading plans to the basis solution. The plot on top compares the initial heat load for each cask, and the plot on bottom compares the time at which each cask meets the transportation requirements. The maximum value of the cask availability dates for each distribution is in 2025.

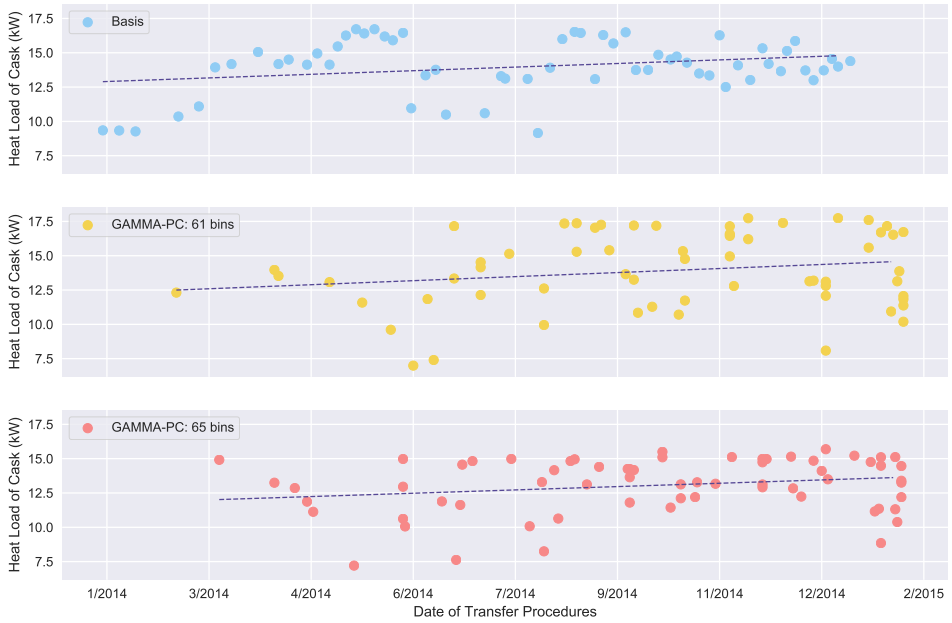


Figure 4.29: Scatter plots comparing the initial cask heat loads for the recommended Zion loading plan and its alternate to the basis solution over the original timeline.

alternate plan happened almost entirely in the top half of the graph. The extra cask loaded every quarter reduced the heat of the hottest casks but did not increase the number of cooler casks below the trendline.

The transportation eligibility dates have fewer distictions among the basis, recommended, and alternate solutions. The biggest difference in Fig. 4.30 is the location of the casks along the $t_{fill,i}$ range. Most of the casks were available for transportation on the transfer date, which makes sense given the number of old assemblies in the pool. The casks with $t_{trans,min}$ dates later than 2020 are governed by the MAGNATRAN loading curve and the presence of higher burnup assemblies. The only difference the loading configuration could make is by grouping the higher burnup assemblies into

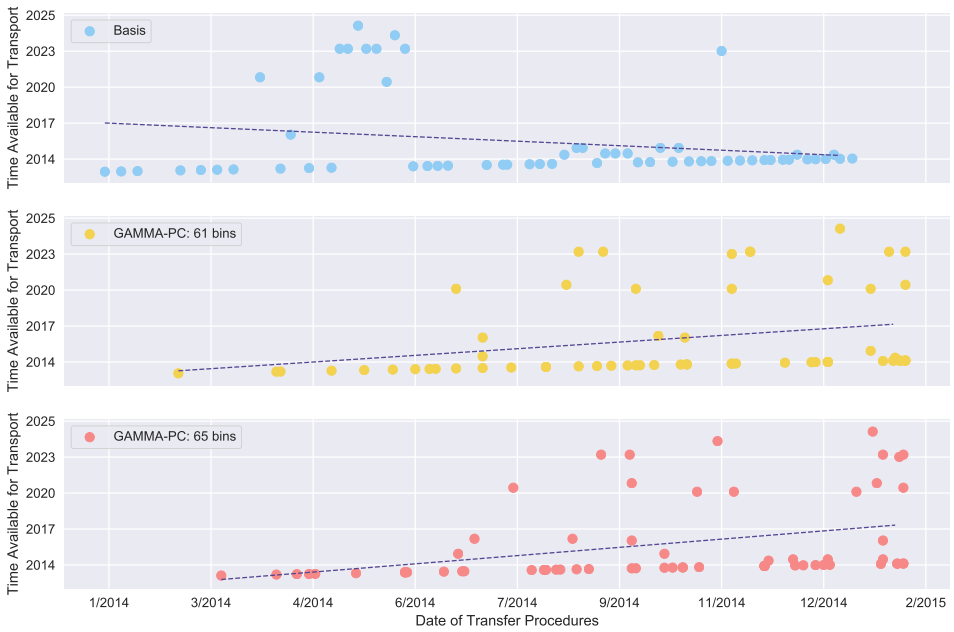


Figure 4.30: Scatter plots comparing the time to transport for the casks in the recommended Zion loading plan and its alternate to the basis solution over the original timeline.

fewer casks, which would adversely affect the second objective without improving the value of the third.

In comparing the GAMMA-PC results to the basis solution, the heat load of the casks has shown the most contrast. The recommended solution dominates the basis using the paradigm developed in Sec. 2 and has a lower overall cask average initial heat load, with and without the weighting system described in Eq. 2.14. However, it also has higher individual initial heat loads than the basis solution.

Figure 4.31 presents heat maps of the hottest cask in the basis solution and in the recommended solution to better illustrate this difference. The hottest cask loaded in the recommended solution includes a wider range of decay heat levels,

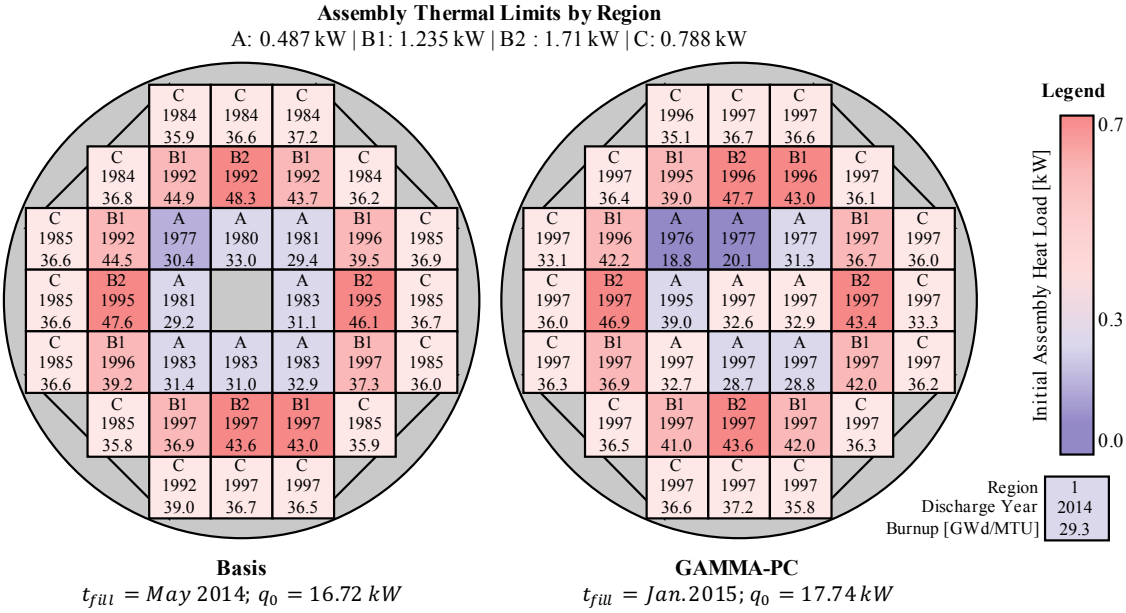
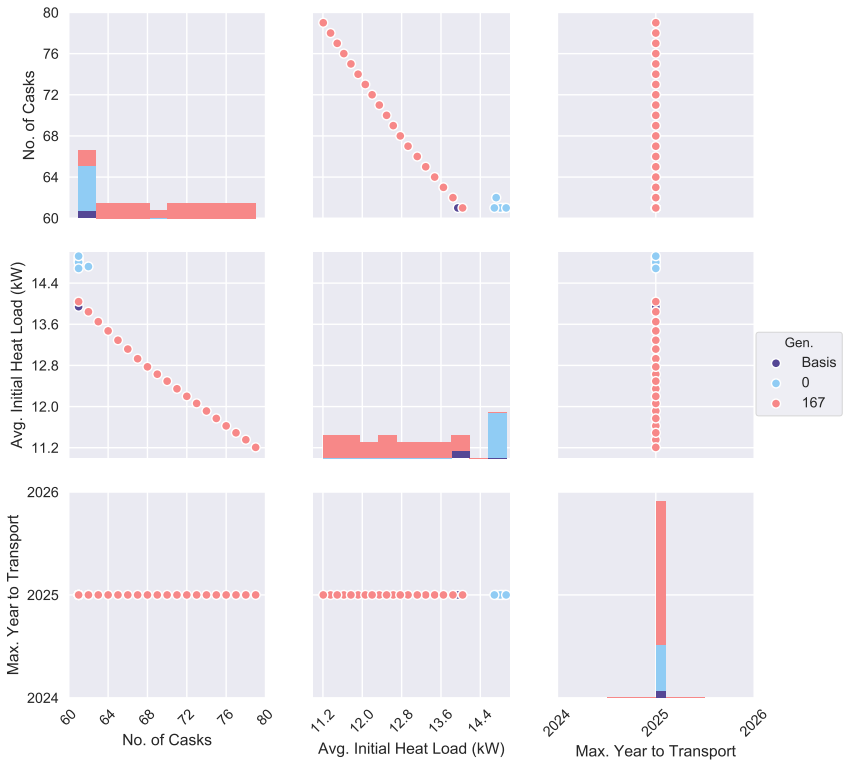


Figure 4.31: Heat maps for the hottest casks in the basis solution and in the recommended GAMMA-PC solution under the original timeline. Both casks are shown using the MAGNASTOR four-zone PWR preferential loading pattern. Note that the assemblies in this graph have only been optimized to the regional loading, and specific placements within the regions are arbitrary.

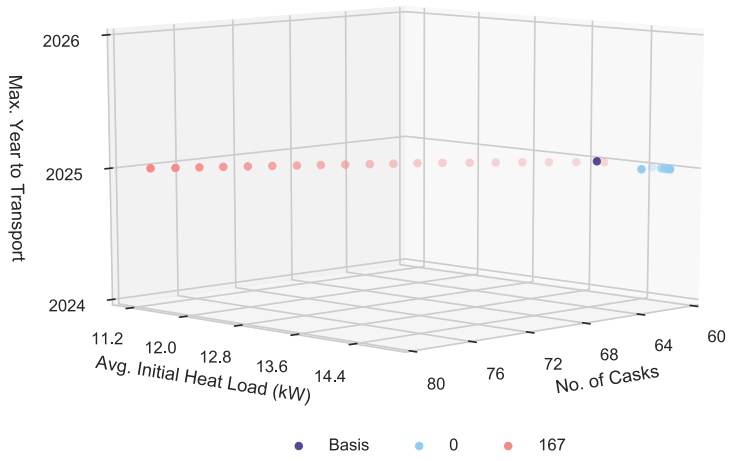
with more assemblies falling on the higher end of the heat scale. The basis solution features a more strategic mix of fuel with a wider range of discharge dates and an empty position. After summing the decay heat in each cask, the hottest basis cask is a full kW cooler than the hottest recommended cask. Under a different optimization paradigm, these two solutions might be nondominated as they exhibit tradeoffs between lowering the average and reducing the maximum initial heat load.

4.3.2 Using a Ten Year Timeframe

In the second optimization of the Zion scenario, the optimization timeline was extended to a 10 year period. ZionSolutions loaded the used fuel in slightly more than a year, which could be a difficult schedule to keep with every cask taking five to



(a)



(b)

Figure 4.32: Scatter plots of the objective space for the Zion dry cask loading problem using the 10 year timeframe. The solutions are plotted on a scatter matrix in (a) and on 3D scatter plot in (b). These graphs show the evolution of the solutions found by GAMMA-PC from the initial generation to the approximate set, found in generation 167. The basis solution is located at $[61, 13.94, 2025]$.

six days to load [123]. The purpose of the extended optimization was to investigate how cask characteristics might change if the fuel had been loaded on a more relaxed timeline. It will be shown that the earlier loading schedule would negatively impact achievable values for the second objective.

Figure 4.32 illustrates the objective space for the Zion problem under the extended timeline. Each dot represents the objective vector for one solution, and the histograms along the diagonal show the distribution of values for each objective function. These figures show the same general trends as was observed under the original timeline: a universal value for the time to transport and an indirect relationship between the number of casks and the average initial heat load. However, with more casks being loaded on earlier dates, the approximate set produced by GAMMA-PC shifted toward higher initial heat loads, which is the expected response given the nature of radioactive decay. The basis solution now dominates the recommended solution at $F = [61, 14.04, 2025]$, although it does not dominate the entire approximate set.

Figure 4.33 presents the distributions of the initial heat loads and the $t_{trans,min,i}$ values under the extended timeline. The GAMMA-PC violins show an evolution toward loading configurations with lower initial heat loads and lower variability in general. The approximate set produced by GAMMA-PC also exhibits a lower cumulative median heat load and median time to transport than the basis solution. Even though the recommended solution is dominated by the basis, the other solutions found by GAMMA-PC might be reasonable alternatives.

Figure 4.34 filters the previous plot to only show the basis, the recommended solution, and the alternate solution from GAMMA-PC. The alternate loading plan in this scenario loads one additional cask for every year of transfer procedures, or 68 in total to cover the seven years corresponding to $F_p = 10$. The distribution for the

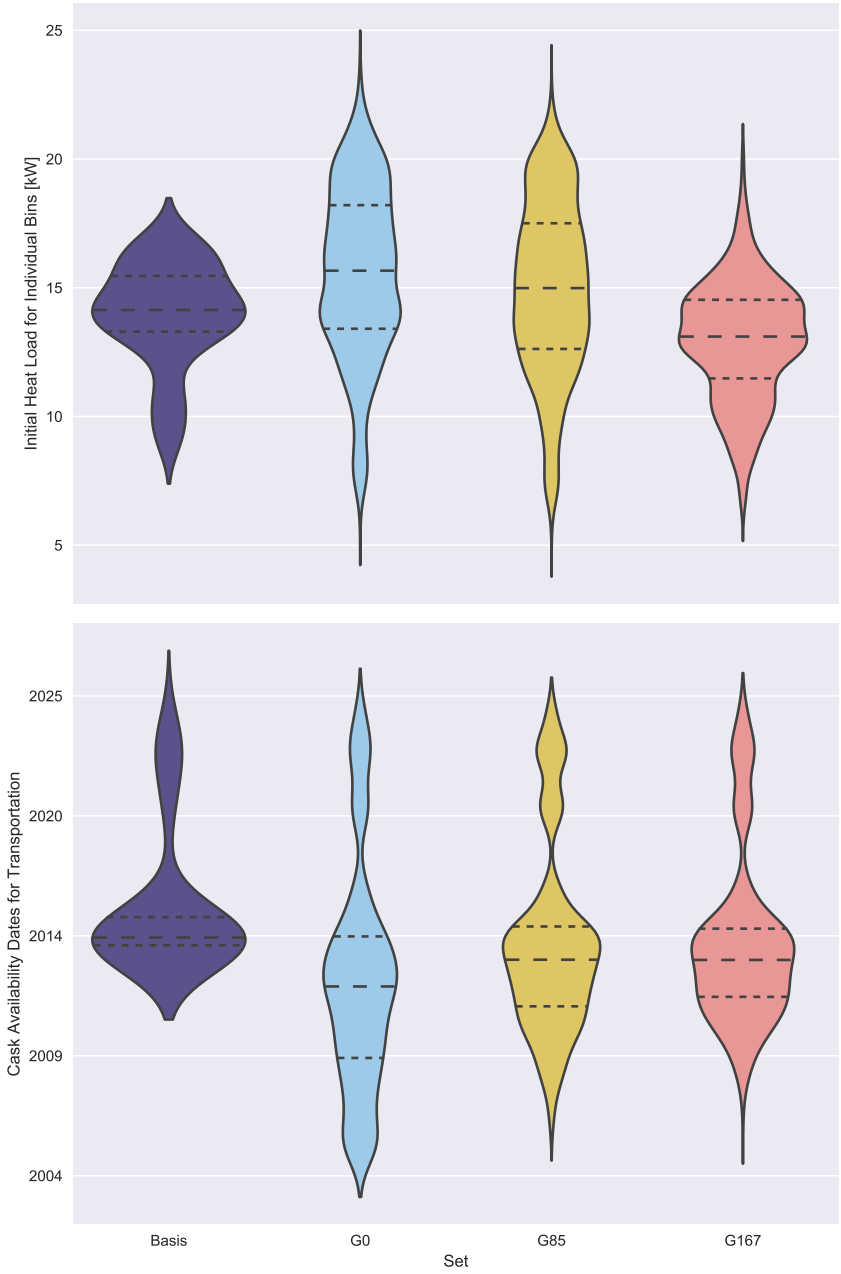


Figure 4.33: Violin plot comparing the cask characteristics of solutions for the Zion dry cask loading problem under the 10 year timeframe. The basis is compared to solutions found by GAMMA-PC during initialization, in generation 85, and in the approximate set (G167). The top plot shows the distribution of the heat loads, and the bottom shows the time at which each cask meets the transportation requirements.

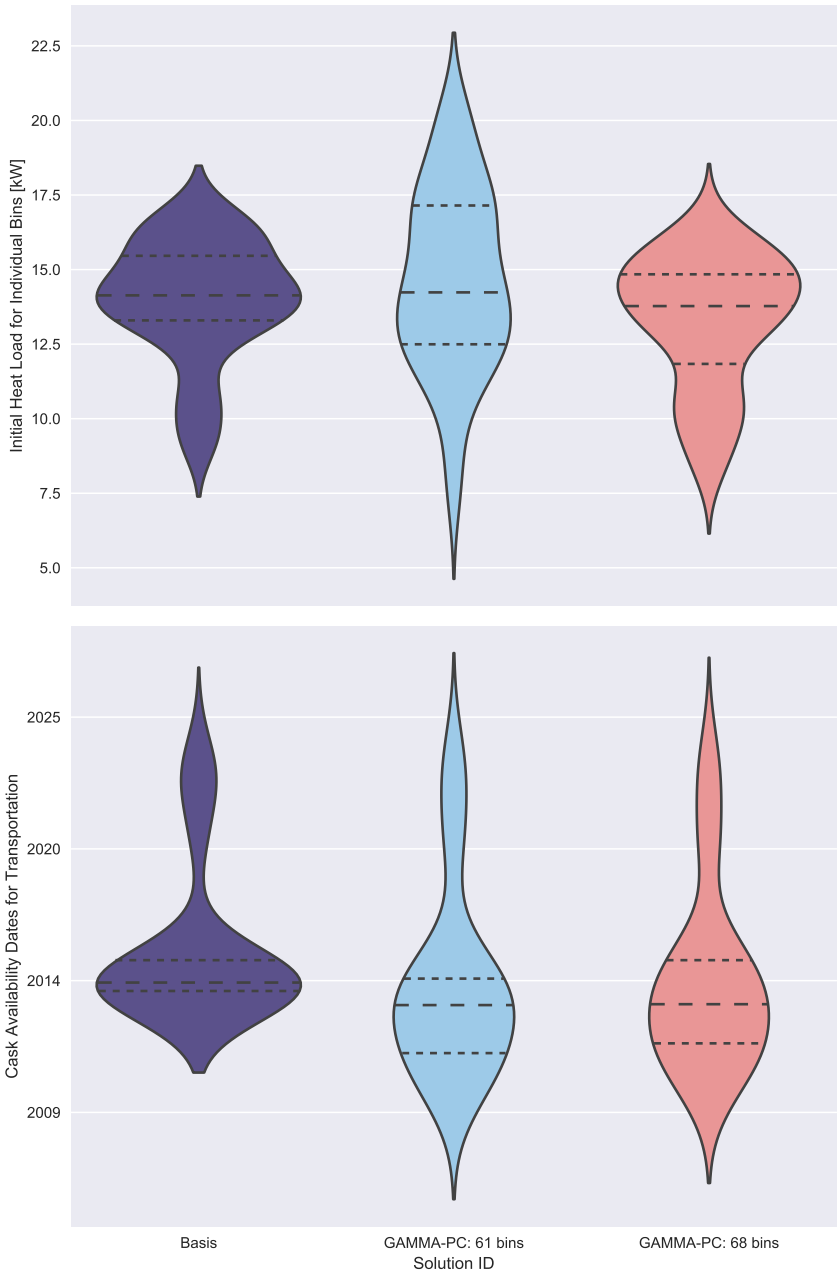


Figure 4.34: Violin plots comparing the cask-specific characteristics of the recommended 10-year Zion loading plan and its alternate to the basis solution. The plot on top compares the distribution of initial cask heat loads, and the plot on bottom compares the time at which each cask meets the transportation requirements.

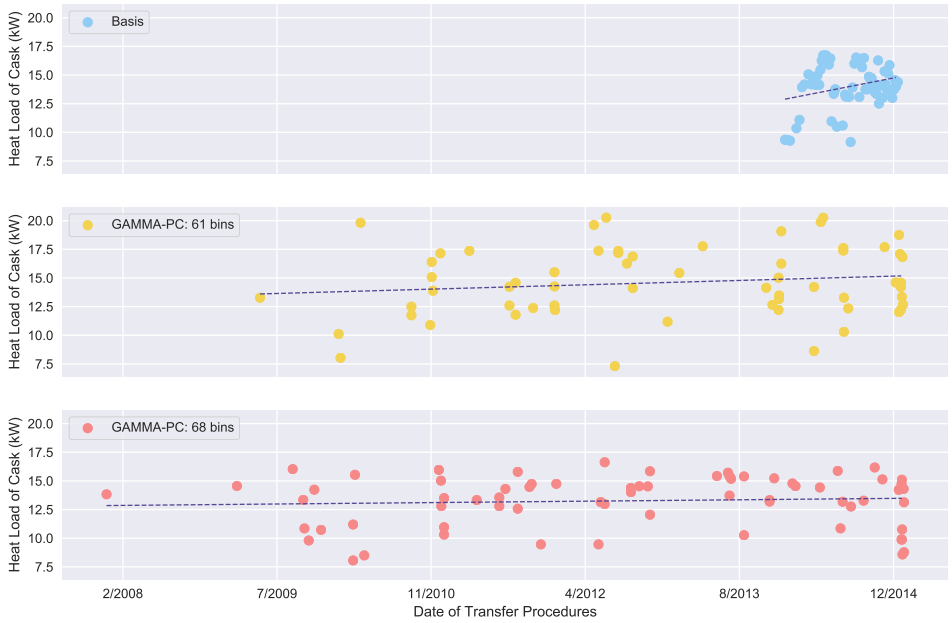


Figure 4.35: Scatter plots comparing the initial cask heat loads for the recommended 10-year Zion loading plan and its alternate to the basis solution over the extended timeline.

heat load of the recommended solution has approximately the same median as the basis, but its top quarter hottest casks are much hotter than the corresponding casks for the basis. On the other hand, the alternate solution exhibits a tigher distribution than the recommended solution and a lower median heat load than the basis. The time to transport distributions for GAMMA-PC both extend earlier than the basis and have lower medians.

Moving the transfer dates earlier in time has been shown to degrade the performance of the recommended solution with respect to the basis. Figure 4.35 shows the initial heat load of the casks as a function of $t_{fill,i}$. The middle plot features hotter casks that were loaded earlier than the original transfer schedule, which is expected

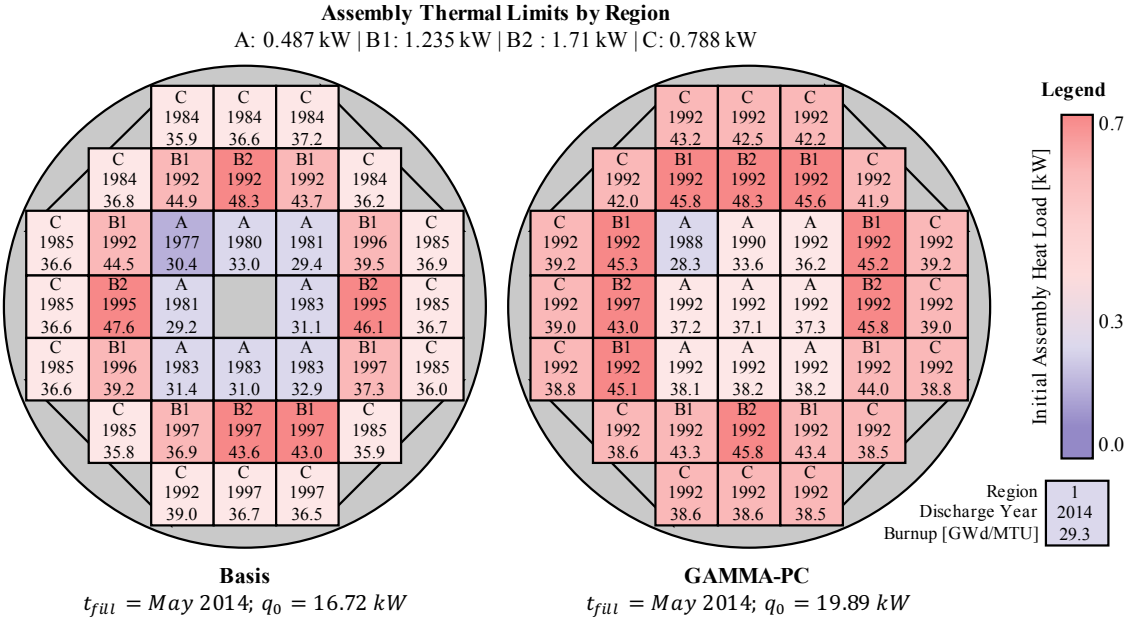


Figure 4.36: Heat maps for the hottest casks loaded in May 2014 for the basis and the recommended GAMMA-PC solutions under the extended timeline. Both casks are shown using the MAGNASTOR four-zone PWR preferential loading pattern. The heat scale does not encompass the maximum heat load limit due to a sizeable margin, and specific placements within the regions are arbitrary.

since used fuel assemblies cool over time. However, it also indicates that a couple of the casks filled within the original schedule are signficantly hotter than would be expected. Figure 4.36 presents the heat map for one of these casks in comparison to the hottest cask for the basis. The map shows a greater concentration of hotter assemblies in the cask with only one assembly cooler than 0.3 kW. This suggests that the recommended solution from GAMMA-PC did not achieve a well-balanced loading configuration.

The variability of heat load values was also seen under the original timeline, albeit to a lesser extent, which suggests that either the packing strategies in GAMMA-PC have room for improvement in terms of balancing the load throughout the casks or

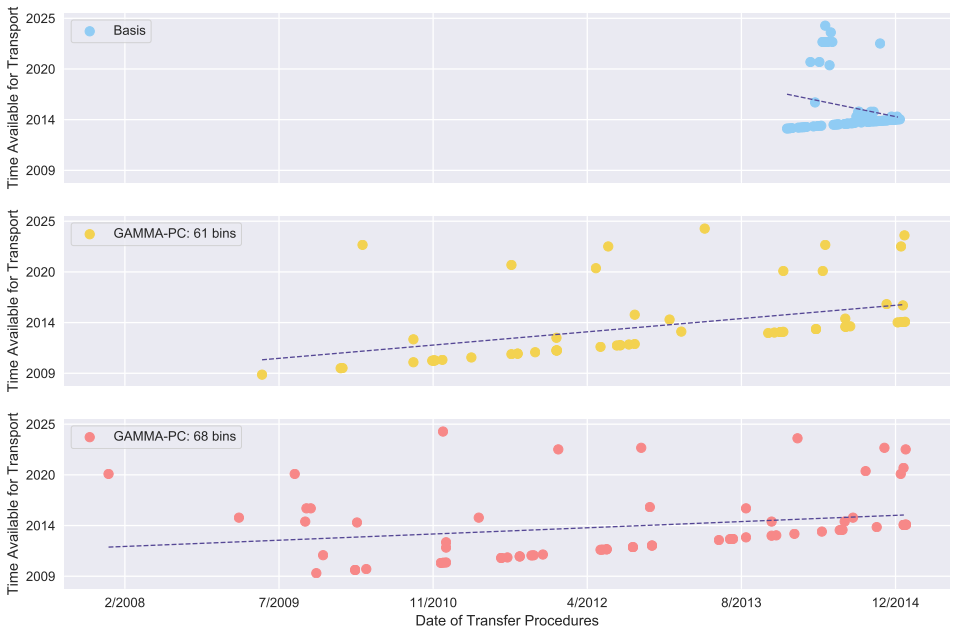


Figure 4.37: Scatter plots comparing the time to transport for the casks in the recommended 10-year Zion loading plan and its alternate to the basis solution over the extended timeline.

the objectives set by the problem formulation do not include a proxy for this quality. Since heat load balance was not an objective defined in the problem formulation, this finding does not suggest that GAMMA-PC performed poorly. Rather GAMMA-PC moved toward solutions that minimized the stated goals, and future optimization studies may want to include balance as an objective. This could be achieved by minimizing the difference between the hottest and coldest casks or by minimizing the standard deviation from the average.

Figure 4.35 does show that the variability of the heat load decreased with an increase in the number of casks. The alternate solution features lower cask heat loads overall and exhibits a fairly uniform pattern throughout the loading schedule.

However, this loading pattern would also leave approximately 10% of the positions open, which could negatively impact the economics of decommissioning.

The impact of the loading configuration on the time to transport has a similar relationship as it did under the original timeline, as shown in Fig. 4.37. Many casks could be transported on the loading day, and the casks that are eligible after 2020 are dependent on the cooling time of the higher burnup assemblies. The low impact on $t_{trans,min}$ suggests that the third objective function is not universally applicable to all ISFSI scenarios as a way to differentiate loading configurations.

4.4 General Trends in the Test Case Scenarios

The results of the optimization cases showed that GAMMA-PC performs well at finding nondominated solutions for its stated objectives. In terms of its performance as an optimization methodology, GAMMA-PC consistently produced a diverse set of solutions that dominated the set of solutions used for comparison. The only case in which the comparison set covered the approximate set produced by GAMMA-PC was the Zion loading problem under the extended timeline, where the basis solution dominated the solution with the minimum number of casks in use. The results of that particular case were more negative for the extended loading schedule than for the performance of GAMMA-PC.

Another consistent finding in the results were scenarios in which the fitness values for the third objective function reduced to a single number independent of the other two objective function values. This was the case when the time to transport was determined on the assembly-level, in which case the loading configuration could not find a lower f_3 value. This may be evidence that future work in this area reframe the third objective to handle the question of transportation differently.

Finally, the analysis of the individual cask characteristics highlighted the fact that

the loading problem paradigm does not directly address balance among the casks. This was particularly demonstrated in the Zion loading problem, where the external solution had been so carefully tuned. Since this was not one of the objectives given to GAMMA-PC, more variability was seen in its cask distributions. Future work might incorporate this as a goal.

5. CONCLUSIONS

To address the evolving needs of dry storage at U.S. nuclear power plants, a new optimization methodology was developed in this research to identify optimal loading configurations for competing objectives relevant to worker safety and long-term transportability:

1. minimizing the number of casks needed to store current and future inventory,
2. minimizing the average initial heat load of those casks, and
3. minimizing the time for the casks to meet transportation guidelines.

The motivation behind this research was to move away from short-term planning strategies, selecting the oldest and coldest used fuel for dry storage, and toward strategies that balance and reduce risk over the lifetime of a site's reactor(s).

The long-term dry cask loading problem was developed as an adaptable paradigm, accomodating different site structures and different cask CoC limits in broadly-defined constraints. It belongs to the class of bin packing problems, which seek to minimize the number of bins used to store a set of items. It is a dynamic, combinatorial problem following the general optimization paradigm shown in Eq. 5.1.

$$\begin{aligned} minimize \quad & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})), \\ s.t. \quad & \mathbf{x} \in (\Omega_{bpp} \cap \Omega_{lc} \cap \Omega_{pool} \cap \Omega_{oper}). \end{aligned} \tag{5.1}$$

The decision vector $\mathbf{x}$ for the loading problem is composed of three arrays: the packing matrix x, the bin array y, and the loading time array t_{fill}. Each row in these arrays correspond to an individual cask being filled, and each column in x corresponds to an individual assembly being loaded. Feasible values for the decision

vector are bounded by bin packing physicality constraints (Ω_{bpp}), assembly selection constraints (Ω_{lc}), spent fuel pool constraints (Ω_{pool}), and operational constraints (Ω_{oper}). Limitations within these general categories are dependent on the specific site being optimized and the cask system chosen. Feasible solutions are evaluated by their performance in each objective function and are compared to other solutions based on their objective vector $\mathbf{F}(x, y, t_{fill})$.

Based on the complexities of the dry cask loading problem, a new metaheuristic algorithm was developed in this research, named the GRASP-enabled adaptive multiobjective memetic algorithm with partial clustering (GAMMA-PC). This novel method embeds greedy randomized adaptive search procedures in the general framework of a multiobjective evolutionary algorithm combined with local search techniques. The packing of the assemblies into casks is handled by greedy heuristics, and the search for new solutions is handled by genetic and local search operators. The crossover operator performs a partial decomposition of the objective space, randomly pairing half of the solutions and selectively pairing the other half based on their local cluster. Application rates for mutation and local search are updated every generation, and the feasibility of new decision vectors is ensured through solution repair methods.

GAMMA-PC was implemented within a Python package, which was useful for the object-oriented nature of the problem. The tool was developed to ensure that its code would be adaptable, maintainable, extensible, and modular. These features will allow future users to optimize different objectives, add new constraints, and create new local search routines. The modularity of the package enables updates to be made separately based on the needed change. The tool also includes a unit testing suite to verify that new changes do not negatively impact other areas of the package.

GAMMA-PC was demonstrated through integration with UNF-ST&DARDS, a

unified database containing information for over 150,000 used fuel assemblies at nuclear facilities around the U.S. [1]. Three sites were optimized under a total of five different scenarios:

- Vermont Yankee Nuclear Power Station using the same cask system and transfer timeframe as the vendor in charge of decommissioning,
- Comanche Peak Nuclear Power Plant using the HI-STORM 100 MPC-32 cask,
- Comanche Peak Nuclear Power Plant using the HI-STORM FW MPC-37 cask,
- Zion Nuclear Power Station using the same cask system and transfer timeframe as the vendor in charge of decommissioning, and
- Zion Nuclear Power Station using an extended ten year transfer timeline.

The performance of GAMMA-PC was evaluated through comparisons of its results with other sets of solutions. For the Vermont Yankee and Comanche Peak scenarios, the GAMMA-PC results were compared to solutions found using only the *First Fit-T* heuristic, which represented oldest and coldest loading configurations. For the Zion cases, the comparison was made against the real loading configuration used by ZionSolutions, LLC.

GAMMA-PC performed well in the test cases, consistently producing a diverse set of solutions that dominated the testing set. The improvement was mainly achieved in the second objective function. For the sites in decommissioning, Vermont Yankee and Zion, the recommended solution from GAMMA-PC reduced the weighted average initial heat load by about 0.2 kW and by 0.1 kW, respectively, when loading fuel over the real-world transfer timeframe. The highest reduction was found in the Comanche Peak scenarios, where the recommended solution reduced the heat load

by 0.76 kW (MPC-32) and by 1.0 kW (MPC-37) when compared to the solution from the testing set with the lowest weighted average initial heat load. In these cases, the higher levels of heat load reduction were the result of applying GAMMA-PC to Comanche Peak's long optimization timeline, compared to the relatively short time frames for Vermont Yankee and Zion.

The results also showed room for improvement in the mathematical formulation of the dry cask loading problem. In four of the five optimization cases, the fitness values in the approximate set for the third objective function reduced to one value independent of the other two objective functions. This single value was the result of the third objective function being partly determined by individual assembly characteristics and partly determined by the loading configuration. It was expected that the values for the third objective would be determined by a mixture of the two aspects, but the results did not support that hypothesis.

5.1 New and Significant Contributions

This research fills a gap in the current literature. Previous used fuel dry storage loading studies have not revealed the mathematical formulation of the problem or have not included the regulatory guidelines for assembly selection in the constraints. This research provided a mathematical framework for the problem, covering longer timeframes than previous programs, and incorporating regulatory limits within the problem architecture. Future dry storage optimization studies can use the formulation as a basis on which to make improvements.

Previous dry loading studies also have not revealed the algorithm used to perform the optimization or have used methods that have limited ability to handle the large, complex search space of the dry cask loading problem. In constrast, GAMMA-PC was developed using suggestions from recent studies on multiobjective evolutionary

algorithms and was validated against other state-of-the-art methods before being applied to the dry cask loading problem (refer to Appendix B).

Finally, this research contributes to studies on bin packing problems. Among the papers published since the first multiobjective bin packing problem, few have featured objective vectors with three or more objectives [103, 155]. The focus here was on the practical application of the class to address an important issue in the field of nuclear engineering, whereas most multiobjective bin packing studies handle ideal bins and items. Moreover, a thorough search of relevant literature was unable to find another study that formulated a dynamic bin packing problem in a similar manner. The dry cask loading problem is different from previous dynamic bin packing problems where items have arrival and departure times, such as those used to model cloud computing [131]. Instead, the assemblies have time-dependent characteristics, wait in the spent fuel pool for some time, and remain in the casks once loaded.

5.2 Recommendations

The results of the five optimization scenarios highlighted some findings for utilities planning loading campaigns.

1. *Consider using GAMMA-PC or another long-term optimization algorithm to improve current loading strategies.* The algorithm could be used as a supplement to current planning procedures, to provide alternative loading configurations for comparison and to evaluate the long-term impact of the chosen short-term campaign plan. It could also be reconfigured as a wrapper or a learning phase for a single-cask loading software, such as CASKLOADER. The results of the optimization scenarios suggested that more diverse loading strategies can improve cask heat loads, in which case identifying assemblies to be saved for future loading cycles would be beneficial.

2. *When using an optimization algorithm such as GAMMA-PC, carefully formulate the objective functions.* The results in Sec. 4 showed that GAMMA-PC found solutions that performed well by the established objective metrics but not as well for other characteristics, such as the variability of the cask heat load. Aligning the objective functions with the planning goals would produce better solutions for decision makers.

3. *Empty positions in canisters should be chosen strategically.* When the total number of assemblies is not divisible by the capacity of the cask, the placement of the empty positions can be used to reduce the heat load of the hottest casks. This benefit was shown in both the Vermont Yankee and Zion cases.

5.3 Future Work

Future work in this area should focus on refining the loading problem statement. The third objective function should be changed to a metric that is solely determined by the selection of assemblies within a cask. In its current form, the third objective function was developed to closely align with the regulatory transportation requirements, which had the unintended consequence of making it highly sensitive to attributes that were independent of the decision vector. Future implementations of the problem should perform preliminary sensitivity analyses on the objective functions to ensure that the competing goals are dependent on the loading configuration.

Future development should also include an objective function focused on the balance of the heat load among the casks. The distributions of the cask initial heat loads and transportation eligibility dates showed variability and was particularly highlighted when compared to the distributions for the real loading of Zion Nuclear Power Station. This result suggests that in its current form, the optimization paradigm did not place priority on this aspect or include a proxy for it in the ob-

jective functions. One approach would be to include an objective to minimize the maximum difference between the hottest and the coldest casks, and another would be to minimize the standard deviation of the initial heat loads.

Any future modifications to the mathematical paradigm would need to be reflected in changes to local search techniques used by GAMMA-PC. Part of the success of the method is that it is built on problem-specific knowledge that guides the search toward more favorable areas of the objective space. Therefore, a new objective function should coincide with the introduction of at least an updated neighborhood structure and possibly a new type of cask modification module in addition to those discussed in Sec. 3.4.

Future research in this area should also incorporate the uncertainty associated with real used fuel systems. While the loading matrix x is binary, the characteristics of the assemblies have some uncertainty associated with them, so the objective functions and the constraints should be updated to deal with the inherent fuzziness of these models. Aspects of GAMMA-PC should also updated to handle the uncertainty. For example, the binary selection procedure would need to be modified to an operator similar to the statistical selection procedure [156].

Core shuffling and reload patterns have been the focus of many optimization studies over the past 20 years, and it is time for the dry cask loading problem to be the subject of formal investigation as well. This research has contributed one of the first in-depth studies in this area. The mathematical paradigm was developed to expand the current treatment of assembly selection and to meet user-defined expectations. GAMMA-PC is a promising new metaheuristic for this task and for dynamic bin packing problems in general.

REFERENCES

[1] K. Banerjee, J. Scaglione, R. LeFebvre, G. Radulescu, and K. Robb, "Streamlining analysis capabilities for SNF management," in *Waste Manage. 2015*. WM Symposia, March 2015.

[2] R. K. Haling, "Operational strategy for maintaining an optimum power distribution throughout life," in *Amer. Nucl. Soc. Topical Meeting, Nucl. Performance of Power-Reactor Cores*. San Francisco, CA: American Nuclear Society, September 1964.

[3] I. Wall and H. Fenech, "The application of dynamic programing to fuel management optimization," *Nucl. Sci. and Eng.*, vol. 22, no. 3, pp. 285 – 297, 1965.

[4] T. O. Sauar, "Application of linear programming to in-core fuel management optimization in light water reactors," *Nucl. Sci. and Eng.*, vol. 46, no. 2, pp. 274 – 283, 1971.

[5] J. S. Miller and N. D. Eckhoff, "A linear programming fuel management model for the HTGR," *Ann. of Nucl. Energy*, vol. 2, no. 9-10, pp. 649 – 656, 1975.

[6] H. Motoda, J. Herczeg, and A. Sesonske, "Optimization of refueling schedule for light-water reactors," *Nucl. Technology*, vol. 25, no. 3, pp. 477 – 496, March 1975.

[7] S. A. Comes and P. J. Turinsky, "Out-of-core fuel cycle optimization for nonequilibrium cycles," *Nucl. Technology*, vol. 83, no. 1, pp. 31–48, 1988.

[8] K. C. Okafor and T. Aldemir, "Construction of linear empirical core models for pressurized water reactor in-core fuel management," *Nucl. Technology*, vol. 81,

no. 3, pp. 381 – 392, June 1988.

[9] J. A. Stillman, Y. A. Chao, and T. J. Downar, "The optimum fuel and power distribution for a pressurized water reactor burnup cycle," *Nucl. Sci. and Eng.*, vol. 103, no. 4, pp. 321 – 333, December 1989.

[10] D. P. Burte and S. G. Vaidya, "Parametrization for optimization of reload patterns for boiling water reactors," *Ann. of Nucl. Energy*, vol. 20, no. 4, pp. 237 – 249, April 1993.

[11] Y. P. Mahlers, "Core loading pattern optimization for pressurized water reactors," *Ann. of Nucl. Energy*, vol. 21, no. 4, pp. 223 – 227, April 1994.

[12] T. Šmuc, D. Pevec, and B. Petrović, "Annealing strategies for loading pattern optimization," *Ann. of Nucl. Energy*, vol. 21, no. 6, pp. 325–336, 1994.

[13] M. A. Feltus, "An extended discharge burnup optimization technique using penn state's fuel management package and CASMO-3/SIMULATE-3," *Ann. of Nucl. Energy*, vol. 22, no. 5, pp. 267 – 274, May 1995.

[14] J. G. Stevens, K. S. Smith, K. R. Rempe, and T. J. Downar, "Optimization of pressurized water reactor shuffling by simulated annealing with heuristics," *Nucl. Sci. and Eng.*, vol. 121, no. 1, pp. 67 – 88, September 1995.

[15] M. D. DeChaine and M. A. Feltus, "Fuel management optimization using genetic algorithms and expert knowledge," *Nucl. Sci. and Eng.*, vol. 124, pp. 188 – 196, 1996.

[16] J. K. Axmann, "Parallel adaptive evolutionary algorithms for pressurized water reactor reload pattern optimizations," *Nucl. Technology*, vol. 119, no. 3, pp. 276 – 290, September 1997.

[17] T. K. Kim and C. H. Kim, "Mixed integer programming for pressurized water reactor fuel-loading-pattern optimization," *Nucl. Sci. and Eng.*, vol. 127, no. 3, pp. 346 – 357, November 1997.

[18] Y. P. Mahlers, "Core loading pattern optimization for research reactors," *Ann. of Nucl. Energy*, vol. 24, no. 7, pp. 509 – 514, May 1997.

[19] A. Yamamoto, "A quantitative comparison of loading pattern optimization methods for in-core fuel management of PWR," *J. of Nucl. Sci. and Technology*, vol. 34, no. 4, pp. 339 – 347, April 1997.

[20] C. Lin, J.-I. Yang, K.-J. Lin, and Z.-D. Wang, "Pressurized water reactor loading pattern design using the simple tabu search," *Nucl. Sci. and Eng.*, vol. 129, no. 1, pp. 61 – 71, April 1998.

[21] J. L. C. Chapot, F. Carvalho Da Silva, and R. Schirru, "A new approach to the use of genetic algorithms to solve the pressurized water reactor's fuel management optimization problem," *Ann. of Nucl. Energy*, vol. 26, no. 7, pp. 641 – 655, 1999.

[22] J. L. François, C. Martín Del Campo, C. C. Cortés, E. Ramírez, and J. Arellano, "Development of an automated system for fuel reload patterns design," *Nucl. Eng. and Design*, vol. 193, no. 1, pp. 13 – 21, September 1999.

[23] B. R. Moore, P. J. Turinsky, and A. A. Karve, "FORMOSA-B: A boiling water reactor in-core fuel management optimization package," *Nucl. Technology*, vol. 126, no. 2, pp. 153–168, 1999.

[24] V. G. Toshinsky, H. Sekimoto, and G. I. Toshinsky, "Multiobjective fuel management optimization for self-fuel-providing LMFBR using genetic algorithms," *Ann. of Nucl. Energy*, vol. 26, no. 9, pp. 783 – 802, June 1999.

[25] A. A. Karve and P. J. Turinsky, "FORMOSA-B: a boiling water reactor in-core fuel management optimization package II," *Nucl. Technology*, vol. 131, no. 1, pp. 48–68, 2000.

[26] A. Yamamoto and H. Hashimoto, "Application of temperature parallel simulated annealing to loading pattern optimizations of pressurized water reactors," *Nucl. Sci. and Eng.*, vol. 136, no. 2, pp. 247 – 257, October 2000.

[27] W. Hongchun, "Pressurized water reactor reloading optimization using genetic algorithms," *Ann. of Nucl. Energy*, vol. 28, no. 13, pp. 1329 – 1341, September 2001.

[28] S. Jagawa, T. Yoshii, and A. Fukao, "Boiling water reactor loading pattern optimization using simple linear perturbation and modified tabu search methods," *Nucl. Sci. and Eng.*, vol. 138, no. 1, pp. 67 – 77, May 2001.

[29] H. C. Lee, H. J. Shim, and C. H. Kim, "Parallel computing adaptive simulated annealing scheme for fuel assembly loading pattern optimization in PWRs," *Nucl. Technology*, vol. 135, no. 1, pp. 39 – 50, July 2001.

[30] Y. Kobayashi and E. Aiyoshi, "Optimization of boiling water reactor loading pattern using two-stage genetic algorithm," *Nucl. Sci. and Eng.*, vol. 142, no. 2, pp. 119 – 139, October 2002.

[31] L. Machado and R. Schirru, "The Ant-Q algorithm applied to the nuclear reload problem," *Ann. of Nucl. Energy*, vol. 29, no. 12, pp. 1455 – 1470, 2002.

[32] A. Erdoan and M. Geçkinli, "A PWR reload optimisation code (XCore) using artificial neural networks and genetic algorithms," *Ann. of Nucl. Energy*, vol. 30, no. 1, pp. 35 – 53, 2003.

[33] E. F. Faria and C. Pereira, "Nuclear fuel loading pattern optimisation using a neural network," *Ann. of Nucl. Energy*, vol. 30, no. 5, pp. 603–613, 2003.

[34] M. Sadighi, S. Setayeshi, and A. A. Salehi, "PWR fuel management optimization using neural networks," *Ann. of Nucl. Energy*, vol. 30, no. 4, p. 511, 2003.

[35] A. Yamamoto, "Application of neural network for loading pattern screening of in-core optimization calculations," *Nucl. Technology*, vol. 144, no. 1, pp. 63 – 75, 2003.

[36] F. Alim and K. Ivanov, "Genetic algorithm development for in-core fuel management," in *Trans. of the Amer. Nucl. Soc.*, vol. 90. Pittsburgh, PA: American Nuclear Society, June 2004, pp. 597 – 598.

[37] ——, "Modeling genetic algorithm operators for loading pattern optimization," in *Trans. of the Amer. Nucl. Soc.*, vol. 91. Washington, DC: American Nuclear Society, November 2004, pp. 756 – 757.

[38] C. Guler, S. Levine, K. Ivanov, J. Svarny, V. Krysl *et al.*, "Development of the VVER core loading optimization system," *Ann. of Nucl. Energy*, vol. 31, no. 7, pp. 747 – 772, May 2004.

[39] J. J. Ortiz and I. Requena, "Using a multi-state recurrent neural network to optimize loading patterns in BWRs," *Ann. of Nucl. Energy*, vol. 31, no. 7, pp. 789–803, 2004.

[40] A. Yamamoto, E. Sugimura, Y. Kitamura, and Y. Yamane, "Simultaneous in-core optimization of PWR tandem cycles," in *Trans. of the Amer. Nucl. Soc.*, vol. 91. Washington, DC: American Nuclear Society, November 2004, pp. 766 – 767.

[41] A. K. Ziver, C. C. Pain, J. N. Carter, C. R. E. de Oliveira, A. J. H. Goddard *et al.*, "Genetic algorithms and artificial neural networks for loading pattern optimisation of advanced gas-cooled reactors," *Ann. of Nucl. Energy*, vol. 31, no. 4, pp. 431 – 457, 2004.

[42] F. Alim and K. Ivanov, "Heuristic rules embedded genetic algorithm for loading pattern optimization," in *Trans. of the Amer. Nucl. Soc.*, vol. 92. San Diego, CA: American Nuclear Society, 2005, pp. 624 – 625.

[43] J. L. François, C. Martín Del Campo, L. B. Morales, and M. A. Palomera, "BWR fuel lattice optimization using scatter search," in *Trans. of the Amer. Nucl. Soc.*, vol. 92. San Diego, CA: American Nuclear Society, June 2005, pp. 615 – 617.

[44] P. M. Keller, "Adaptively constrained multiobjective genetic algorithms for in-core fuel management optimization," in *Trans. of the Amer. Nucl. Soc.*, vol. 92. San Diego, CA: American Nuclear Society, 2005, pp. 610 – 611.

[45] Q. B. Do, G. Roh, and H. Choi, "Optimization of a refueling simulation for a CANDU reactor by using an evolutionary algorithm," in *Trans. of the Amer. Nucl. Soc.*, vol. 94. Reno, NV: American Nuclear Society, June 2006, pp. 388 – 389.

[46] T. Wang and Z. Xie, "A hybrid optimization method for loading pattern search," in *Trans. of the Amer. Nucl. Soc.*, vol. 94. Reno, NV: American Nuclear Society, June 2006, pp. 390 – 391.

[47] A. Castillo, J. J. Ortiz, J. L. Montes, and R. Perusquía, "Fuel loading and control rod patterns optimization in a BWR using tabu search," *Ann. of Nucl. Energy*, vol. 34, no. 3, pp. 207–212, 2007.

[48] J. L. François, C. Martín Del Campo, L. B. Morales, and M. A. Palomera, "Development of a scatter search optimization algorithm for boiling water reactor fuel lattice design," *Nucl. Sci. and Eng.*, vol. 155, no. 3, pp. 367 – 377, March 2007.

[49] S. M. H. Hadavi, "Risk-based, genetic algorithm approach to optimize outage maintenance schedule," *Ann. of Nucl. Energy*, vol. 35, no. 4, pp. 601–609, 2007.

[50] H. Hernandez and G. I. Maldonado, "Application of simulated annealing optimization to recycle minor actinides in a BWR lattice," in *Trans. of the Amer. Nucl. Soc.*, vol. 96. Boston, MA: American Nuclear Society, June 2007, pp. 771 – 773.

[51] P. M. Keller, K. R. Rempe, G. Anton, J. L. Eller, and L. C. James, "Development of a parallelized incore optimization tool utilizing multiobjective genetic algorithms and a licensed nodal reactor simulator," in *Trans. of the Amer. Nucl. Soc.*, vol. 96. Boston, MA: American Nuclear Society, 2007, pp. 618 – 619.

[52] Y. Kim, K. J. Chang, and J. M. Noh, "Optimization of axial fuel shuffling strategy in a block-type VHTR," in *Trans. of the Amer. Nucl. Soc.*, vol. 97. Washington, DC: American Nuclear Society, November 2007, pp. 406 – 407.

[53] A. M. M. d. Lima, R. Schirru, F. C. d. Silva, M. D. Machado, and J. A. C. C. Medeiros, "Study of heuristics in ant system for nuclear reload optimization," in *Proc. of the INAC 2007 Int. Nucl. Atlantic Conf.*, vol. 39, no. 44, International Nuclear Atlantic Conference. Associacao Brasileira de Energia Nuclear, October 2007.

[54] T. K. Park, H. C. Lee, H. K. Joo, and C. H. Kim, "Improvement of screening efficiency in loading pattern optimization by simulated annealing," in *Trans. of*

the Amer. Nucl. Soc., vol. 96. Boston, MA: American Nuclear Society, June 2007, pp. 578 – 579.

[55] F. Alim, K. Ivanov, and S. H. Levine, "New genetic algorithms (GA) to optimize PWR reactors: Part III: The haling power depletion method for in-core fuel management analysis," *Ann. of Nucl. Energy*, vol. 35, no. 1, pp. 121 – 131, January 2008.

[56] A. H. Fadaei and S. Setayeshi, "LONSA as a tool for loading pattern optimization for VVER-1000 using synergy of a neural network and simulated annealing," *Ann. of Nucl. Energy*, vol. 35, no. 10, pp. 1968 – 1973, 2008.

[57] A. M. M. d. Lima, R. Schirru, F. C. d. Silva, and J. A. C. C. Medeiros, "A nuclear reactor core fuel reload optimization using ant colony connective networks," *Ann. of Nucl. Energy*, vol. 35, no. 9, pp. 1606 – 1612, 2008.

[58] T. K. Park, H. K. Joo, and C. H. Kim, "Multi-objective fuel loading optimization employing discontinuous penalty functions in simulated annealing," in *Trans. of the Amer. Nucl. Soc.*, vol. 98. Anaheim, CA: American Nuclear Society, June 2008, pp. 47 – 48.

[59] A. A. De Moura Meneses, M. D. Machado, and R. Schirru, "Particle swarm optimization applied to the nuclear reload problem of a pressurized water reactor," *Progress in Nucl. Energy*, vol. 51, no. 2, pp. 319 – 326, March 2009.

[60] M. Waintraub, R. Schirru, and C. M. N. A. Pereira, "Multiprocessor modeling of parallel particle swarm optimization applied to nuclear engineering problems," *Progress in Nucl. Energy*, vol. 51, no. 6, pp. 680 – 688, August 2009.

[61] G. Žerovnik, L. Snoj, and M. Ravnik, "Optimization of spent nuclear fuel filling in canisters for deep repository," *Nucl. Sci. and Eng.*, vol. 163, no. 2, pp.

183–190, 2009.

[62] A. A. De Moura Meneses, L. M. Gambardella, and R. Schirru, "A new approach for heuristics-guided search in the in-core fuel management optimization," *Progress in Nucl. Energy*, vol. 52, no. 4, pp. 339 – 351, May 2010.

[63] S. Ishida and H. Sekimoto, "Finding the best fuel assemblies shuffling scheme of ADS for MA transmutation using dynamic programming," *Nucl. Eng. and Design*, vol. 240, no. 10, pp. 3645 – 3653, 2010.

[64] N. Shaukat, S.-U.-I. Ahmad, A. Majeed, N. Ahmad, and B. Mohsin, "Optimization of core reload pattern for PARR-1 using evolutionary techniques," *Nucl. Eng. and Design*, vol. 240, no. 10, pp. 2831 – 2835, 2010.

[65] E. Zio and R. Bazzo, "Multiobjective optimization of the inspection intervals of a nuclear safety system: A clustering-based framework for reducing the pareto front," *Ann. of Nucl. Energy*, vol. 37, no. 6, pp. 798 – 812, June 2010.

[66] R. Hays and P. J. Turinsky, "BWR in-core fuel management optimization using parallel simulated annealing in FORMOSA-B," *Progress in Nucl. Energy*, vol. 53, no. 6, pp. 600–606, 2011.

[67] D. J. Kropaczek, "COPERNICUS: A multi-cycle optimization code for nuclear fuel based on parallel simulated annealing with mixing of states," *Progress in Nucl. Energy*, vol. 53, no. 6, pp. 554 – 561, August 2011.

[68] A. Norouzi, A. Zolfaghari, A. H. Minuchehr, and F. Khoshahval, "An enhanced integer coded genetic algorithm to optimize PWRs," *Progress in Nucl. Energy*, vol. 53, no. 5, pp. 449 – 456, 2011.

[69] I. M. S. de Oliveira and R. Schirru, "Swarm intelligence of artificial bees applied to in-core fuel management optimization," *Ann. of Nucl. Energy*, vol. 38, no. 5,

pp. 1039–1045, 2011.

[70] J. J. Ortiz-Servin, J. A. Castillo, and D. A. Pelta, "BWR fuel cycle optimization using neural networks," *Nucl. Eng. and Design*, vol. 241, no. 9, pp. 3729 – 3735, 2011.

[71] M. H. da Silva and R. Schirru, "Optimization of nuclear reactor core fuel reload using the new quantum PBIL," *Ann. of Nucl. Energy*, vol. 38, no. 2 - 3, pp. 610 – 614, February 2011.

[72] M. H. da Silva, R. Schirru, and A. M. M. de Lima, "QACOAlpha applied to the nuclear reactor core fuel reload optimization," *Progress in Nucl. Energy*, vol. 53, no. 1, pp. 80 – 85, January 2011.

[73] P. V. Tsvetkov, T. G. Lewis III, S. K. Lakshmipathy, A. M. Ougouag, and F. Venneri, "Performance-constrained reload pattern searches for deep burn HTR hexagonal block systems," in *Trans. of the Amer. Nucl. Soc.*, vol. 104. Hollywood, FL: American Nuclear Society, June 2011, pp. 688 – 689.

[74] R. D. S. Yadav and H. P. Gupta, "Optimization studies of fuel loading pattern for a typical pressurized water reactor (PWR) using particle swarm method," *Ann. of Nucl. Energy*, vol. 38, no. 9, pp. 2086–2095, 2011.

[75] S. Carlos, A. Sanchez, S. Martorell, and J.-F. Villanueva, "Particle swarm optimization of safety components and systems of nuclear power plants under uncertain maintenance planning," *Advances in Eng. Software*, vol. 50, no. 1, pp. 12 – 18, August 2012.

[76] R. D. Hays, "Stochastic optimization for nuclear facility deployment scenarios," Ph.D. Dissertation, North Carolina State University, Raleigh, NC,

2012. [Online]. Available: repository.lib.ncsu.edu/ir/bitstream/1840.16/8085/1/etd.pdf

[77] C. Lin and B.-F. Lin, "Automatic pressurized water reactor loading pattern design using ant colony algorithms," *Ann. of Nucl. Energy*, vol. 43, no. 0, pp. 91–98, 2012.

[78] S. Liu and J. Cai, "Studies of fuel loading pattern optimization for a typical pressurized water reactor (PWR) using improved pivot particle swarm method," *Ann. of Nucl. Energy*, vol. 50, no. 0, pp. 117–125, 2012.

[79] M. Aghaie, T. Nazari, A. Zolfaghari, A. Minuchehr, and A. Shirani, "Investigation of PWR core optimization using harmony search algorithms," *Ann. of Nucl. Energy*, vol. 57, pp. 1 – 15, 2013.

[80] D. J. Kropaczek, M. Asgari, and M. Mahgerefteh, "Method for addressing hybrid-equilibrium loading constraints within the COPERNICUS multi-cycle nuclear fuel optimization code," in *Trans. of the Amer. Nucl. Soc.*, vol. 108. American Nuclear Society, June 2013, pp. 729–731.

[81] S. Levine, T. Blyth, and K. Ivanov, "Optimizing PWR low leakage cores with genetic algorithms and other techniques," in *Trans. of the Amer. Nucl. Soc.*, vol. 108. Atlanta, GA: American Nuclear Society, June 2013, pp. 716 – 719.

[82] C.-D. Wang and C. Lin, "Automatic boiling water reactor control rod pattern design using particle swarm optimization algorithm and local search," *Nucl. Eng. and Design*, vol. 255, no. 0, pp. 273–279, 2013.

[83] S. S. Arshi, A. Zolfaghari, and S. M. Mirvakili, "A multi-objective shuffled frog leaping algorithm for in-core fuel management optimization," *Computer Physics Communications*, vol. 185, no. 10, pp. 2622 – 2628, 2014.

[84] R. W. Carlsen, M. J. Gidden, and P. P. Wilson, "Deployment optimization with the CYCLUS fuel cycle simulator," in *Trans. of the Amer. Nucl. Soc.*, vol. 111. American Nuclear Society, November 2014, pp. 241–244.

[85] A. Hedayat, "Developing a practical optimization of the refueling program for ordinary research reactors using a modified simulated annealing method," *Progress in Nucl. Energy*, vol. 76, pp. 191 – 205, September 2014.

[86] F. Khoshahval, A. Zolfaghari, and H. Minuchehr, "A new method for multi-objective in core fuel management optimization using biogeography based algorithm," *Ann. of Nucl. Energy*, vol. 73, no. 0, pp. 294–303, 2014.

[87] F. Khoshahval, A. Zolfaghari, H. Minuchehr, and M. R. Abbasi, "A new hybrid method for multi-objective fuel management optimization using parallel PSO-SA," *Progress in Nucl. Energy*, vol. 76, pp. 112 – 121, 2014.

[88] C. Lin and Y.-H. Chen, "The max-min ant system and tabu search for pressurized water reactor loading pattern design," *Ann. of Nucl. Energy*, vol. 71, no. 0, pp. 388–398, 2014.

[89] T. K. Park, H. G. Joo, and C. H. Kim, "Multicycle fuel loading pattern optimization by multiobjective simulated annealing employing adaptively constrained discontinuous penalty function," *Nucl. Sci. and Eng.*, vol. 176, no. 2, pp. 226 – 239, February 2014.

[90] M. H. da Silva and R. Schirru, "A self-adaptive quantum PBIL method for the nuclear reload optimization," *Progress in Nucl. Energy*, vol. 74, pp. 103 – 109, July 2014.

[91] A. Zameer, S. M. Mirza, and N. M. Mirza, "Core loading pattern optimization of a typical two loop 300 mwe PWR using simulated annealing (SA), novel

crossover genetic algorithms (GA) and hybrid GA(SA) schemes," *Ann. of Nucl. Energy*, vol. 65, pp. 122 – 131, 2014.

[92] J. P. Câmara Augusto, A. D. Santos Nicolau, and R. Schirru, "PSO with dynamic topology and random keys method applied to nuclear reactor reload," *Progress in Nucl. Energy*, vol. 83, pp. 191 – 196, August 2015.

[93] A. A. De Moura Meneses and R. Schirru, "A cross-entropy method applied to the in-core fuel management optimization of a pressurized water reactor," *Progress in Nucl. Energy*, vol. 83, pp. 326 – 335, 2015.

[94] N. J. Hill and G. T. Parks, "Pressurized water reactor in-core nuclear fuel management by tabu search," *Ann. of Nucl. Energy*, vol. 75, no. 0, pp. 64–71, 2015.

[95] K. E. Ottinger and G. I. Maldonado, "BWROPT: A multi-cycle BWR fuel cycle optimization code," *Nucl. Eng. and Design*, vol. 291, pp. 236 – 243, June 2015.

[96] H. Park and H. G. Joo, "The optimization of subgroup levels with the simulated annealing method," in *Trans. of the Amer. Nucl. Soc.*, vol. 113, no. 1. American Nuclear Society, October 2015, pp. 1232 – 1235.

[97] N. Poursalehi, A. Zolfaghari, and A. Minuchehr, "A novel optimization method, effective discrete firefly algorithm, for fuel reload design of nuclear reactors," *Ann. of Nucl. Energy*, vol. 81, pp. 263 – 275, 2015.

[98] J. Su, C. Wang, S. Wang, H. Yu, and Y. Chen, "A hybrid optimization method for loading pattern search used in COSINE package," in *Trans. of the Amer. Nucl. Soc.*, vol. 113, no. 1. American Nuclear Society, October 2015, pp. 321 – 323.

[99] N. Ayoobian and M. Mohsendokht, "Multi-objective optimization of maintenance programs in nuclear power plants using genetic algorithm and sensitivity index decision making," *Ann. of Nucl. Energy*, vol. 88, pp. 95 – 99, February 2016.

[100] J. Hou, S. Qvist, R. Kellogg, and E. Greenspan, "3D in-core fuel management optimization for breed-and-burn reactors," *Progress in Nucl. Energy*, vol. 88, pp. 58 – 74, April 2016.

[101] E. B. Schlünz, P. M. Bokov, R. H. Prinsloo, and J. H. Van Vuuren, "A unified methodology for single- and multiobjective in-core fuel management optimisation based on augmented Chebyshev scalarisation and a harmony search algorithm," *Ann. of Nucl. Energy*, vol. 87, pp. 659 – 670, January 2016.

[102] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evol. Computation*, vol. 6, no. 2, pp. 182 – 197, 2002.

[103] N. Dahmani, F. Clautiaux, S. Krichen, and E.-G. Talbi, "Self-adaptive metaheuristics for solving a multi-objective 2-dimensional vector packing problem," *Appl. Soft Computing*, vol. 16, pp. 124 – 136, 2014.

[104] BRC, "Disposal subcommittee report to the full commission," Blue Ribbon Commission on American's Nuclear Future, Tech. Rep., 2012. [Online]. Available: https://curie.ornl.gov/content/disposal-subcommittee-report-full-commission

[105] NRC. (2015) U.S. independent spent fuel storage installations. U.S. Nuclear Regulatory Commission. [Online]. Available: http://pbadupws.nrc.gov/docs/ML1507/ML15078A414.pdf

[106] ——, "Generic environmental impact statement for continued storage of spent nuclear fuel," U.S. Nuclear Regulatory Commission, Office of Nuclear Material Safety and Safeguards, Washington, DC, Tech. Rep. NUREG-2157, 2014.

[107] IAEA, "Optimization strategies for cask design and container loading in long term spent fuel storage," International Atomic Energy Agency, Nuclear Fuel Cycle and Materials Section, Vienna, Austria, Report IAEA-TECDOC-1523, 2006.

[108] ——, "Survey of wet and dry spent fuel storage," International Atomic Energy Agency, Nuclear Fuel Cycle and Materials Section, Vienna, Austria, Tech. Rep. IAEA-TECDOC-1100, 1999.

[109] *Code of Federal Regulations*. U.S. National Archives and Records Administration, July 2015, no. 10, "Energy," Part 72.214 (10 CFR 72.214). [Online]. Available: http://www.nrc.gov/reading-rm/doc-collections/cfr/part072/part072-0214.html

[110] D. R. Leduc, "Dry storage of used fuel transition to transport," Savannah River National Laboratory, Department of Energy, Tech. Rep. FCRD-UFD-2012-000253, 2012.

[111] NRC, *Interim Staff Guidance - 8, Revision 3: Burnup Credit in the Criticality Safety Analyses of PWR Spent Fuel in Transportation and Storage Casks*. U.S. Nuclear Regulatory Commission, Division of Spent Fuel Storage and Transportation, 2012. [Online]. Available: https://www.nrc.gov/reading-rm/doc-collections/isg/isg-8R3.pdf

[112] ——, *Certificate of Compliance No. 1032 for the HI-Storm Flood/Wind System*. U.S. Nuclear Regulatory Commission, July 2011, no. ML111950103. [Online]. Available: http://pbadupws.nrc.gov/docs/ML1119/ML111950103.html

[113] J. C. Wagner, "Evaluation of burnup credit for accommodating PWR spent nuclear fuel in high-capacity cask designs," in *Proc. of 7th Int. Conf. on Nucl. Criticality Safety (ICNC2003)*. Tokai, Ibaraki, Japan: American Nuclear Society, October 2003, pp. 684 – 689.

[114] NRC, *Interim Staff Guidance - 8, Revision 0: Limited Burnup Credit in the Criticality Safety Analyses of PWR Spent Fuel in Transport and Storage Casks*. U.S. Nuclear Regulatory Commission, Spent Fuel Project Office, 1999. [Online]. Available: https://curie.ornl.gov/system/files/documents/not%20yet%20assigned/ISG-8R0.pdf

[115] J. Scaglione, D. E. Mueller, J. C. Wagner, and W. J. Marshall, "An approach for validating actinide and fission product burnup credit criticality safety analyses–criticality (keff) predications," U.S. Nuclear Regulatory Commission, Oak Ridge National Laboratory, Oak Ridge, TN, Tech. Rep. NUREG-7109, 2012. [Online]. Available: https://www.nrc.gov/docs/ML1211/ML12116A128.pdf

[116] C. V. Parks, J. C. Wagner, D. E. Mueller, and I. C. Gauld, "Development of technical basis for burnup credit regulatory guidance in the united states," in *Proc. of the 16th Int. Symp. on the Packaging and Transport of Radioactive Materials (PATRAM 2010)*, 2010.

[117] J. C. Wagner and D. E. Mueller, "Updated evaluation of burnup credit for accommodating PWR spent nuclear fuel in high-capacity cask designs," in *2005 NCSD Topical Meeting: Integrating Criticality Safety into the Resurgence of Nuclear Power*. Knoxville, TN: American Nuclear Society, Nuclear Criticality Safety Division, September 2005.

[118] J. C. Wagner, J. L. Peterson, D. E. Mueller, J. C. Gehin, A. Worrall *et al.*, "Categorization of used nuclear fuel inventory in support of a comprehensive national nuclear fuel cycle strategy," Oak Ridge National Laboratory, Department of Energy, Tech. Rep. ORNL/TM-2012/308, 2012.

[119] B. McLeod, "Calculation method for the projection of future spent nuclear fuel discharges," Bechtel SAIC Company, LLC for the Office of Civilian Radioactive Waste Management, Tech. Rep. TDR-WAT-NU-000002 REV 02, 2005.

[120] D. Knott and C. Oyarzun, "Description of the shuffle design software MARLA," *Progress in Nucl. Energy*, vol. 53, no. 6, pp. 607–617, 2011.

[121] EPRI, "Program on technology innovation: A quantitative radiological risk analysis of the U.S. once-through nuclear fuel cycle," Electric Power Research Institute, Palo Alto, CA, Report 3002000807, 2013.

[122] ——, "2014 research portfolio: Used fuel and high-level waste management (QA)," Electric Power Research Institute, Tech. Rep., 2014. [Online]. Available: http://mydocs.epri.com/docs/Portfolio/P2014/2014_41-03-01.pdf

[123] W. J. Szymczak and F. N. Williams, "Zionsolutions dry cask storage project / supporting fuel transfer," in *Trans. of the Amer. Nucl. Soc.*, vol. 111. American Nuclear Society, November 2014, pp. 74 – 76.

[124] D.-K. Cho and J. Jeong, "Effectiveness of source term optimization for higher disposal density of spent fuels in a deep geological repository," *Ann. of Nucl. Energy*, vol. 71, no. 0, pp. 125–129, 2014.

[125] M. Kazimi, E. J. Moniz, C. W. Forsberg, G. Apostolakis, P. Hejvlar *et al.*, *The Future of the Nuclear Fuel Cycle*. Boston, MA: Massachusetts Institute of Technology, 2011, no. ISBN = 978-0-9828008-4-3.

[126] K. Banerjee, J. Scaglione, and R. LeFebvre, "Integrated data and analysis tool for used nuclear fuel management," in *Trans. of the Amer. Nucl. Soc.*, vol. 111. American Nuclear Society, November 2014, pp. 338 – 341.

[127] K. Yancey and P. V. Tsvetkov, "Variability in spent fuel inventory data," *Progress in Nucl. Energy*, vol. 81, no. 0, pp. 184 – 195, 2015.

[128] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan *et al.*, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evol. Computation*, vol. 1, no. 1, pp. 32 – 49, 2011.

[129] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. d. Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. on Evol. Computation*, vol. 7, no. 2, pp. 117 – 132, 2003.

[130] D. S. Liu, K. C. Tan, S. Y. Huang, C. K. Goh, and W. K. Ho, "On solving multiobjective bin packing problems using evolutionary particle swarm optimization," *European J. of Operational Research*, vol. 190, no. 2, pp. 357 – 382, 2008.

[131] Y. Li, X. Tang, and W. Cai, "Dynamic bin packing for on-demand cloud resource allocation," *IEEE Trans. on Parallel and Distributed Syst.*, vol. 27, no. 1, pp. 157–170, 2016.

[132] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. New York: J. Wiley & Sons, 1990, ch. 8. Bin-packing problem, pp. 221–245.

[133] S. Soares, C. H. Antunes, and R. Araújo, "Comparison of a genetic algorithm and simulated annealing for automatic neural network ensemble development," *Neurocomputing*, vol. 121, no. 0, pp. 498–511, 2013.

[134] H. Ishibuchi, Y. Hitotsuyanagi, N. Tsukamoto, and Y. Nojima, "Implementation of multiobjective memetic algorithms for combinatorial optimization problems: A knapsack problem case study," *Stud. in Computation Intell.*, vol. 171, pp. 27 – 49, 2009.

[135] EPRI, "Impacts associated with transfer of spent nuclear fuel from spent fuel storage pools to dry storage after five years of cooling, revision 1," Electric Power Research Institute, Palo Alto, CA, Tech. Rep. 1025206, August 2012.

[136] E. A. Schneider, M. R. Deinert, and K. B. Cady, "Cost analysis of the us spent nuclear fuel reprocessing facility," *Energy Economics*, vol. 31, no. 5, pp. 627–634, 2009.

[137] S. O. Hansson, K. Lilieqvist, K. E. Björnberg, and M. V. Johansson, "Time horizons and discount rates in Swedish environmental policy: Who decides and on what grounds?" *Futures*, vol. 76, pp. 55 – 66, 2016.

[138] K. Yancey and P. V. Tsvetkov, "Quantification of U.S. spent fuel inventories in nuclear waste management," *Ann. of Nucl. Energy*, vol. 72, no. 0, pp. 277–285, 2014.

[139] NRC, "Standard review plan for transportation packages for spent nuclear fuel," U.S. Nuclear Regulatory Commission, Washington, DC, Tech. Rep. NUREG-1617, March 2000.

[140] ——, "Standard review plan for transportation packages for MOX spent nuclear fuel," U.S. Nuclear Regulatory Commission, Washington, DC, Tech. Rep. NUREG-1617 Supplement 1, September 2005.

[141] S. Butenko and P. M. Pardalos, *Numerical Methods and Optimization: An Introduction*. Boca Raton, FL: CRC Press, Taylor Francis Group, LLC, 2014.

[142] C. V. Parks, "Annual progress report: Data and analysis for spent nuclear fuel transport and storage in burnup credit casks," Oak Ridge National Laboratory, Electric Power Research Institute, Oak Ridge, TN, Tech. Rep., November 2005.

[143] A. M. Chwatal and S. Pirkwieser, "Solving the two-dimensional bin-packing problem with variable bin sizes by greedy randomized adaptive search procedures and variable neighborhood search," in *Int. Conf. on Computer Aided Syst. Theory*. Las Palmas de Gran Canaria, Spain: Springer-Verlag Berlin Heidelberg, February 2011, pp. 456–463.

[144] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *J. of Global Optimization*, vol. 6, pp. 109–133, 1995.

[145] Y. J. Cao, L. Jiang, and Q. H. Wu, "An evolutionary programming approach to mixed-variable optimization problems," *Appl. Math. Modelling*, vol. 24, pp. 931 – 942, 2000.

[146] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with Gaussian process model," *IEEE Trans. on Evol. Computation*, vol. 14, no. 3, pp. 456–473, 2010.

[147] E. S. Nicoară, "Performance measures for multi-objective optimization algorithms," *Buletinul Universităţii Petrol–Gaze din Ploieşti., Seria Matematică-Informatică-Fizică*, vol. 59, no. 1, pp. 19–28, 2007.

[148] NRC. (2017, February) Vermont Yankee nuclear power station. U.S. Nuclear Regulatory Commission. [Online]. Available: https://www.nrc.gov/info-finder/decommissioning/power-reactor/vermont-yankee.html

[149] WNN, "Holtec completes main work at Vermont Yankee," *World Nucl. News*, August 2017. [Online]. Available: http://www.world-nuclear-news.org/

WR-Holtec-completes-main-work-at-Vermont-Yankee-14081701.html

[150] NRC. (2017, February) Comanche Peak nuclear power plant, unit 1. U.S. Nuclear Regulatory Commission. [Online]. Available: https://www.nrc.gov/info-finder/reactors/cp1.html

[151] ——. (2017, February) Comanche Peak nuclear power plant, unit 2. U.S. Nuclear Regulatory Commission. [Online]. Available: https://www.nrc.gov/info-finder/reactors/cp2.html

[152] Holtec, "Safety analysis report on the HI-STAR 190 package," Holtec International, Marlton, NJ, Tech. Rep. HI-2146214, August 2015.

[153] C. W. Pennington, "NAC's modular, advanced generation, nuclear all-purpose storage (MAGNASTOR) system: New generation multipurpose spent fuel storage for global application," in *14th Int. Symp. on the Packaging and Transportation of Radioactive Materials (PATRAM 2004)*. Berlin, Germany: Bundesanstalt für Materialforschung und -prüfung, Institute of Nuclear Materials Management, and International Atomic Energy Agency, September 2004.

[154] NRC, *Draft Certificate of Compliance No. 9356 for the Model No. MAGNATRAN Transport Cask Package.* U.S. Nuclear Regulatory Commission, March 2013, no. ML13093A233. [Online]. Available: https://www.nrc.gov/docs/ML1309/ML13093A233.pdf

[155] K. S. Rao and P. S. Thilagam, "Heuristics based server consolidation with residual resource defragmentation in cloud data centers," *Future Generation Computer Syst.*, vol. 50, no. C, pp. 87 – 98, September 2015.

[156] L. L. Hay, C. E. Peng, T. Suyan, and L. Juxin, "Application of evolutionary algorithms for solving multi-objective simulation optimization problems," in

Multi-Objective Memetic Algorithms, C. K. Goh, K. C. Tan, and Y.-S. Ong, Eds. Singapore: Springer Berlin Heidelberg, 2009, pp. 91 – 110.

[157] L. Ke, Q. Zhang, and R. Battiti, "Hybridization of decomposition and local search for multiobjective optimization," *IEEE Trans. on Cybernetics*, vol. 44, no. 10, pp. 1808 – 1820, 2014.

[158] C. K. Goh, K. C. Tan, D. S. Liu, and S. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *European J. of Operational Research*, vol. 202, pp. 42 – 54, 2010.

[159] K. C. Tan, E. F. Khor, T. H. Lee, and R. Sathikannan, "An evolutionary algorithm with advanced goal and priority specification for multi-objective optimization," *J. of Artificial Intell. Research*, vol. 18, pp. 183 – 215, 2003.

[160] E. Silva, J. F. Oliveira, and G. Wäscher, "2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems," *European J. of Operational Research*, vol. 237, pp. 846 – 856, March 2014.

[161] S. Anily, J. Bramel, and D. Simchi-Levi, "Worst-case analysis of heuristics for the bin packing problem with general cost structures," *Operations Research*, vol. 42, no. 2, pp. 287–298, 1994.

[162] O. D. Lara and M. A. Labrador, "A multiobjective ant colony-based optimization algorithm for the bin packing problem with load balancing," in *2010 IEEE Congress on Evol. Computation (CEC)*. Barcelona, Spain: IEEE Computation Intelligence Society, September 2010, pp. 1–8.

[163] M. Sathe, O. Schenk, and H. Burkhart, "Solving bi-objective many-constraint bin packing problems in automobile sheet metal forming processes," in *Proc. of*

Evol. Multi-Criterion Optimization: 5th Int. Conf., EMO 2009, M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds. Nantes, France: Springer Berlin Heidelberg, April 2009, pp. 246–260.

[164] C. K. Dartey, J. W. Finley, and R. R. Thulin, *Process for Preparing Chocolate Chip Cookies Containing Low Melting Fat and Product*, U.S. Patent No. 4,722,849, issued date Feb. 2, 1988.

[165] N. Nitin and M. V. Karwe, "Heat transfer coefficient for model cookies in a turbulent multiple jet impingement system," in *Transport Phenomena in Food Process.*, J. Welti-Chanes, J. F. Velez-Ruiz, and G. V. Barbosa-Cánovas, Eds. Boca Raton, FL: CRC Press, Taylor Francis Group, LLC, 2002, ch. 24, pp. 357 – 376.

[166] Y. A. Cengel, *Heat Transfer: A Practical Approach*, 2nd ed. New York: Mcgraw-Hill, 2002, no. ISBN-10: 0072458933.

[167] F. A. Kulacki and S. C. Kennedy, "Measurement of the thermophysical properties of common cookie cough," *J. of Food Sci.*, vol. 43, pp. 380 – 384, 1978.

[168] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms – a comparative case study," in *Parallel Problem Solving from Nature (PPSN V)*, A. E. Eiben and et. al., Eds. Amsterdam, The Netherlands: Springer, September 1998, Book Section, pp. 292 – 301.

[169] S. F. Adra and P. J. Flemin, "Diversity management in evolutionary many-objective optimization," *IEEE Trans. on Evol. Computation*, vol. 15, no. 2, pp. 183 – 195, 2011.

[170] V. Grunert da Fonseca, C. M. Fonseca, and A. O. Hall, "Inferential performance assessment of stochastic optimisers and the attainment function," in *Proc. of*

Evol. Multi-Criterion Optimization: First Int. Conf., EMO 2001, E. Zitzler, L. Thiele, K. Deb, and C. A. Coello Coello, Eds. Zurich, Switzerland: Springer Berlin Heidelberg, March 2001, pp. 213–225.

[171] C. M. Fonseca, A. P. Guerreiro, M. López-Ibáñez, and L. Paquete, "On the computation of the empirical attainment function," in *Proc. of Evol. Multi-Criterion Optimization: 6th Int. Conf., EMO 2011*, R. H. Takahashi, K. Deb, E. F. Wanner, and S. Greco, Eds. Ouro Preto, Brazil: Springer Berlin Heidelberg, April 2011, pp. 106–120.

[172] K. D. Lee and S. Hubbard, *Data Structures and Algorithms with Python*, ser. Undergraduate Topics in Computer Science. Luther College, Decorah, IA: Springer International Publishing Switzerland, 2015, no. ISBN-10: 3319130714.

[173] H. Abdi, *Encyclopedia of Measurements and Statistics*. Thousand Oaks, CA: SAGE Publications, Inc., 2007, ch. The Bonferroni and Sidak Corrections for Multiple Procedures.

[174] G. Ochoa, J. Walker, M. Hyde, and T. Curtois, "Adaptive evolutionary algorithms and extensions to the hyflex hyper-heuristic framework," in *Parallel Problem Solving from Nature - PPSN XII: 12th Int. Conf., Proc., Part II*, C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia *et al.*, Eds. Taormina, Italy: Springer Berlin Heidelberg, September 2012, pp. 418–427.

[175] C. M. Fonseca, J. D. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," in *Evol. Multi-Criterion Optimization: First Int. Conf., EMO 2005*, Guanajauto, Mexico, 2005.

[176] F. Satterthwaite, "An approximate distribution of estimates of variance components," *Biometrics Bulletin*, vol. 2, no. 6, pp. 110 – 114, 1946.

APPENDIX A

BIN PACKING BENCHMARK DEVELOPMENT

During the development of GAMMA-PC, its performance was validated and improved by the use of benchmark scenarios. Given the narrow range of previous research into the optimization of dry cask loading patterns, a set of problem-specific benchmark scenarios does not exist. Since the problem belongs to the general class of bin packing problems, more idealized benchmarks were established to aid in development. This section describes how the benchmarks mirror the dry cask loading problem, the optimization methodologies used for comparison, and the lessons learned from their performance in the first benchmark problem. The full evaluation of GAMMA-PC in these benchmark scenarios is given in Appendix B.

A.1 Benchmark Problem Formulation

For the benchmarks to be successful, they needed to reflect the characteristics of the dry cask loading problem while reducing its complexity. To achieve this, two benchmark scenarios were developed: a static case and a dynamic case. Both were formulated with three objectives to simulate similar objective spaces.

A.1.1 Static Problem Modifications

The static problem reduces the dry cask loading case to a standard bin packing problem, sorting idealized objects into larger bins. Each object has a weight and a height associated with it, and the bins are assigned weight and height limits. The goal of this problem is to minimize the number of bins needed while also minimizing the average weight and the maximum used height of the bins. The first objective function Eq. 2.11 can be used without modification. The other two are simplified analogies

of Eqs. 2.12 and 2.15 that test an algorithm's handling of means and maximums in the objective space.

The static problem is only constrained by physical reality. The "no replacement" constraint applies here, and each bin must not be filled beyond the weight or height capacities. The problem uses the standard bin packing problem decision variable format.

A.1.2 Dynamic Problem Modifications

The dynamic benchmark scenario was developed to be a model of the dry cask loading problem, sharing its unique characteristics on a much smaller scale. It uses the same decision variable as the dry cask loading problem and uses the daily operation of a bakery as a metaphor. The problem focuses on baking a set of n cookies. It is assumed that n is large enough that the oven can't bake them all at the same time, so the cookies come out of the oven in batches. Before being placed into bins, or cookie boxes, the cookies must cool off to avoid moisture buildup. However, similar to spent fuel pools, the cooling racks can only hold a limited amount of cookies. Therefore, the boxes must start being filled before the end of the cookie baking session. The characteristics of the assemblies also evolve over time, so not every cookie cools down at the same rate, explained by the presence of flavor particles, such as chocolate chips.

To mirror the cask loading problem, the cookies are sorted into boxes with three goals:

1. minimizing the number of boxes used,

2. minimizing the average initial heat of a box,

3. and minimizing the maximum time until the containers can be moved into the

storefront.

The second objective was formulated by assuming that the cookies are placed in a pattern that makes heat transfer interaction between them negligible. This enabled the assumption that the heat in a box is a linear sum of the convective heat produced by each cooling cookie. The cookies cool exponentially, which makes them a good model for the radioactive decay of nuclides in used fuel assemblies. In the third objective, it was assumed that the box is ready to be moved once the total heat is equivalent to each cookie in a full box being within 5°C of room temperature. The time to move is calculated with the modified regula-falsi method, similar to Eq. 2.15.

The packing constraints were modified for time dependency. The"no replacement" and box capacity constraints were applied, as well as a physicality constraint representing the capacity of the cooling rack. The mathematical formulation of both the static and dynamic benchmark problems are developed in Appendix B.

A.2 Optimization Methodologies for Comparison

Four optimization methodologies were chosen for comparison and validation of GAMMA-PC:

- Nondominated Sorting Genetic Algorithm II (NSGA-II) [102],
- Multiobjective Evolutionary Particle Swarm Optimization (MOEPSO) [130],
- Multiobjective Memetic Algorithm (MOMA) [134],
- and Memetic Algorithm Based on Decomposition (MOMAD) [157].

These methodologies have previously been applied to combinatorial problems and represent a variety of techniques. NSGA-II was chosen as one of the most popular MOEAs used to validate new algorithms, such as in [158]. MOEPSO was chosen

because it was specifically designed for multiobjective bin packing problems, and MOMA was chosen to represent a memetic-version of NSGA-II. Finally, MOMAD was chosen since it is one of the newest memetic algorithms and features a flexible framework.

A.2.1 NSGA-II

NSGA-II was developed by Deb et. al. in 2002 to address criticisms of the original NSGA approach, which were the high computational complexity, the lack of elitism, and the need to specify a sharing parameter to ensure diversity [102]. To fix these issues, the authors implemented a fast sorting algorithm to select the parent population of every generation based on Pareto dominance and developed a more general niche operator to encourage diversity along the Pareto Front. Since its publication, the algorithm has become the basis for the majority of MOEAs [128].

The general flow of NSGA-II is shown in Fig. A.1. The initial population is generated randomly, and the first round of selection is performed using a standard binary tournament with elitism [102]. For the bin packing problems developed here, single-point crossover is performed for all generations, as well as a single mutation that consists of swapping the order of two genes in the chromosome representation. These evolutionary functions are performed at the rates listed in Table A.1. Any new solution in the next generation is encoded into matrix representation before being evaluated and archived.

After the initial generation, the breeding pool is selected based on the fast non-dominated sorting algorithm laid out in [102]. This algorithm uses the current and previous generations as candidates, sorts them into domination fronts, and then fills the breeding population sequentially in order of nondomination rank. Once the algorithm reaches a front that has more members than is required to reach completion,

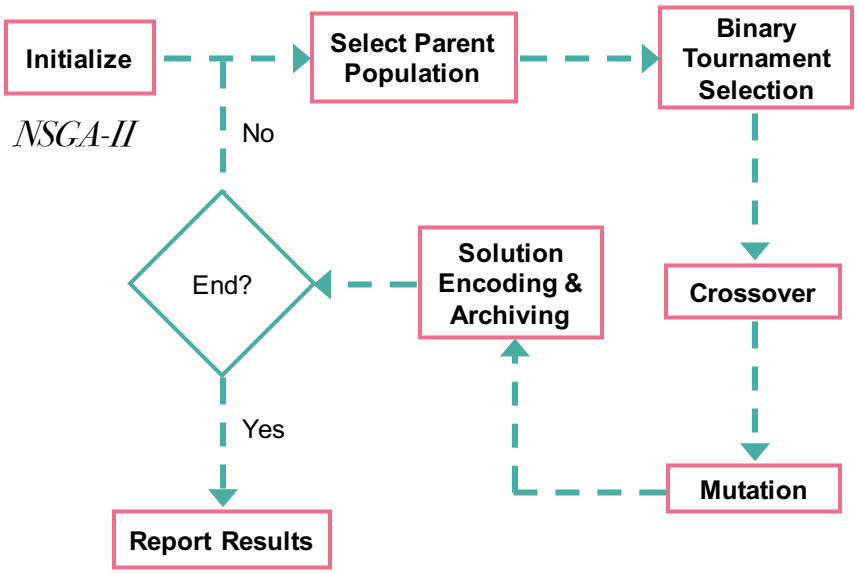


Figure A.1: High level flowchart of NSGA-II algorithm.

Table A.1: NSGA-II Benchmark Settings.

Generation Size	100
Function Evaluations	25,000
Crossover Probability	0.90
Mutation Probability	0.30

the members within that front are added in reverse order to the breeding pool based each solution's niche value. This value is determined by the crowding distance assignment (cda). This niche operator indicates how close a solution is to the other solutions in its domination front inside the objective space, so adding solutions with higher cda values promotes diversity in the breeding pool.

A.2.2 MOEPSO

MOEPSO was developed by Liu et. al. in 2007 to solve multiobjective bin packing problems [130]. Their paper was the first to present a mathematical formulation of a bi-objective bin packing problem and the first to apply particle swarm optimization (PSO) to a bin packing problem. PSO is a population-based stochastic optimization method inspired by the behavior of swarming animals, such as birds or fish [128]. The general way it operates is that the particles in the population, or the swarm, evolve over time, learning from their own history and from the swarm's combined knowledge. The authors chose to use PSO due to evidence of a higher convergence speed for multiobjective optimization. They also incorporated a mutation operator in their method to combine evolutionary computation with PSO concepts. Their results showed that MOEPSO performed better in almost all of their tests than either a standard MOEA or multiobjective PSO alone.

The algorithmic flow of MOEPSO is shown in Fig. A.2. The calculation initializes the swarm by creating chromosome representations using a completely random, a completely sorted, or a partially-random/partially-sorted combination of item indices [130]. Table A.2 gives the swarm size and the probablities of the way each particle's chromosome representation is initialized. The solutions are encoded into both matrix representation and the variable length representation before moving into the main loop.

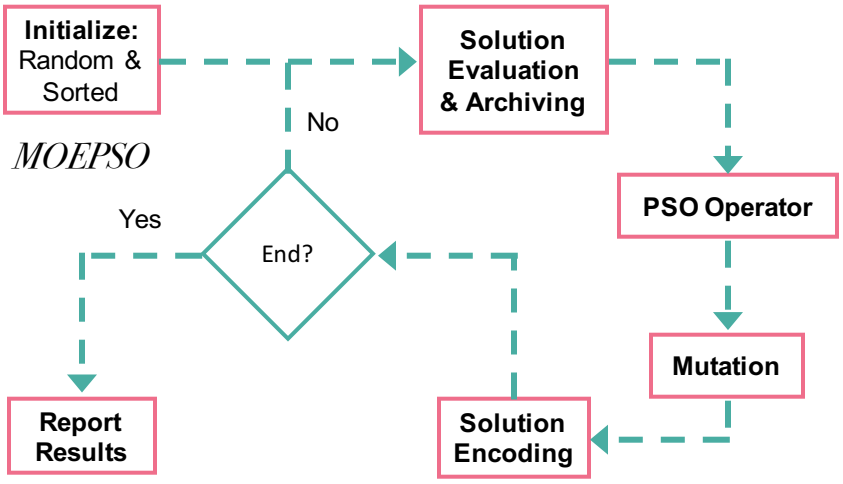


Figure A.2: High level flowchart of MOEPSO algorithm.

Table A.2: MOEPSO Benchmark Settings.

Swarm Size	500
Function Evaluations	25,000
Fixed Size of Global Best Archive	50
Initialization: Probability of Random Sequence	0.50
Initialization: Probability of Random & Sorted Sequence	0.25
Initialization: Probability of Sorted Sequence	0.25

As a modified PSO, MOEPSO maintains two archives: the "personal best" archive, which keeps track of each individual's best solution, and the "global best" archive, which keeps track of the nondominated solutions found by the swarm [130]. The solution evaluation and archiving step shown in Fig. A.2 performs this function. The personal best solution is updated anytime the algorithm finds a solution for a particle that dominates the previous personal best. The global best archive is updated at every generation to include nondominated solutions and is truncated to the pre-determined fixed size using the dynamic sharing scheme [159].

To adapt to the bin packing problem, the PSO operator in MOEPSO adds a best bin into a particle's solution and repacks the others [130]. The best bin is the most-filled bin of a solution that is selected randomly between the particle's personal best or one of the solutions in the global best archive. The bin is added to the particle using the variable length representation, and the other items not in the best bin are gathered into a partial chromosome. The x- and y-matrices are then repacked using the encoding strategy.

The mutation portion of MOEPSO randomly selects between three bin-specific mutation functions [130]. The first operation is to partially swap the contents of two random bins in a solution. The second merges the two least-filled bins together. The third splits a random bin into two separate bins. The mutation operations are limited enough in scope that the modifications to a solution happen in the variable length and matrix representations without needing to use the encoding strategy. However, after these changes have been made, the chromosome representation needs to be updated for the next round. The solution encoding step in Fig. A.2 does this.

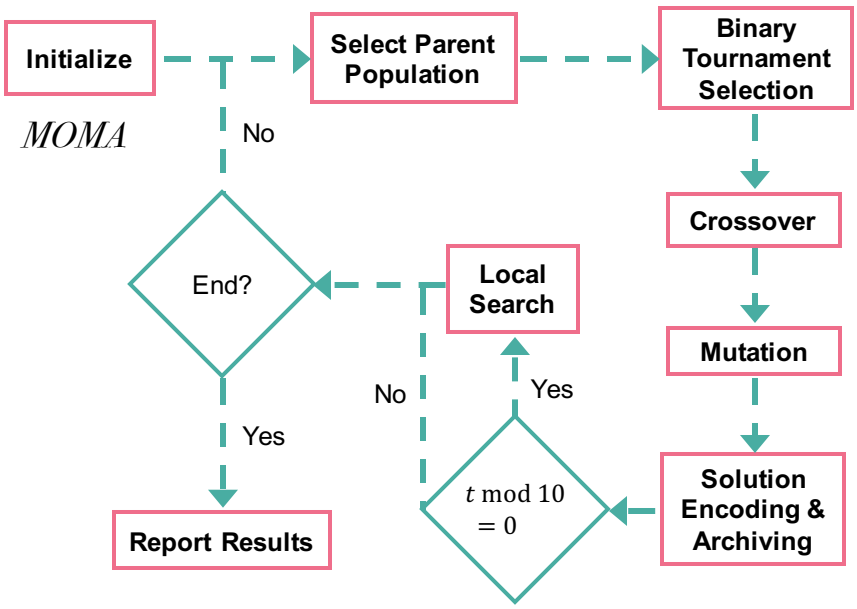


Figure A.3: High level flowchart of MOMA algorithm.

A.2.3 MOMA

MOMA is a memetic version of NSGA-II that was developed by Ishibuchi et. al. in 2009 to demonstrate the impact of varying certain parameters related to local search [134]. They applied the algorithm to multiobjective knapsack problems, which are another type of combinatorial problem. Their experiments considered aspects such as the frequency of local search, the choice of initial solutions for local search, the termination condition of local search, and the handling of infeasible solutions. The results from the experiments showed that for hybridization with local search to improve a method, the local search method needs to incorporate problem-specific knowledge.

Figure A.3 shows the algorithmic flow of MOMA. As MOMA is based on NSGA-II, it shares the same basic structure with the addition of the local search function. The settings for MOMA are given in Table A.3. The additional decision function in

Table A.3: MOMA Benchmark Settings.

Generation Size	100
Function Evaluations	25,000
Crossover Probability	0.80
Mutation Probability	0.30
Local Search Probability	0.0 to 0.2
Local Search Size	10

the figure indicates that the local search is only performed every 10 generations. This incorporates the result that showed similar performance regardless if the frequency were every one, five, or ten generations [134]. Local search is applied to solutions in the breeding pool with a given probablity, and the search size determines the number of neighbors that are identified during one search. The local search probablity specified in Table A.3 shows a range of values because the authors found that MOMA worked best when the probability increased linearly from 0 to 0.2. They also found that it performed best when the product of the probability and the search size stayed within the region from 1.0 to 2.0, so this adaptation used a search size of 10 neighbors.

A change from the original algorithm was that this version of MOMA uses Pareto local search instead of the weighted sum fitness approach [134]. Therefore, if a neighbor is nondominated by the original solution, it's added to the next generation as a potential member of the next parent population. The neighbors are identified with the help of one function from a set of four: one that swaps the placement of two items between bins and three similar to the MOEPSO mutation functions. This modification incorporates the authors' suggestion that local search should use problem-specific information to be successful. In the case of a bin packing problem,

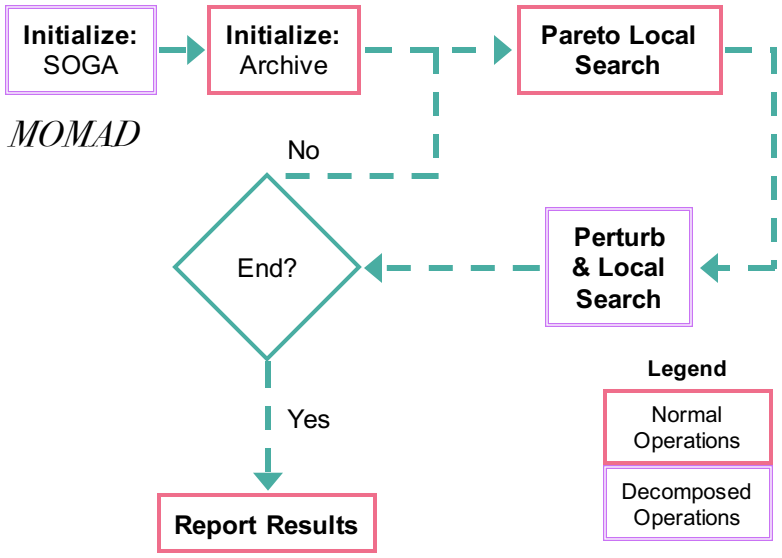


Figure A.4: High level flowchart of MOMAD algorithm.

bins are more useful than the information contained in the chromosome representation.

A.2.4 MOMAD

MOMAD was developed by Ke et. al. in 2014 to combine aspects of evolutionary algorithms, decomposition approaches, and Pareto local search methods [157]. It is one of the latest memetic algorithms and presents a flexible framework for future use. The authors applied MOMAD to a multiobjective traveling salesman problem and a multiobjective knapsack problem. Their results showed promise for solving combinatorial problems with MOMAD.

Figure A.4 presents a high-level view of MOMAD. Normal operations involving the whole multiobjective problem are indicated by a solid pink outline, and decomposed operations are indicated by the double-lined purple outline. MOMAD is initialized by decomposing the multiobjective problem into a number of single-objective

Table A.4: MOMAD Benchmark Settings with specifications for the single-objective (SO) genetic algorithm.

Number of SO Functions	N or 600 (if $N > 600$)
Function Evaluations	25,000
SO Population Size	5
No. of SO Generations	5
SO Crossover Probability	0.90
SO Mutation Probability	0.30
Local Search Probability	1.0
Pareto Local Search Size	50
Subproblem Local Search Size	25

functions [157]. This is done by generating a set of random weight vectors whose size is determined by the number of objectives and whose sum must equal 1. Then, each single-objective function is defined by the cross product of the weight vector and the multiobjective vector. A single-objective optimization method is applied to each subproblem to find an ideal solution for the initial generation. In this application, a genetic algorithm was used with characteristics listed in Table A.4. The population size and the number of generations completed in this stage are small due to the many single-objective functions considered and the limited number of function evaluations.

After the single-objective optimization finishes, the archives are initiated [157]. Three populations are maintained during calculations: P_L, P_P, and P_E. The first population keeps track of the current solutions to the single-objective subproblems, and the second lists the solutions designated for Pareto local search. The P_E population keeps track of all of the nondominated solutions throughout the calculations.

Set P_L is initiated by solving the single-objective subproblems. Both P_P and P_E are initiated by determining the nondominated solutions in P_L.

The main loop of MOMAD consists of Pareto local searches followed by subproblem local searches [157]. The Pareto local search is applied to each of the solutions in P_P with the search size given Table A.4. The P_L and P_E sets are updated as necessary, and after each search, the solution is removed from P_P. In the next step, each solution in P_L is perturbed to generate a new solution. Local search is applied to that new solution, comparing it with one neighbor at a time until a better solution is found, then continuing the search with that solution until the search size is reached. If the search results in a better solution to the subproblem, P_L is updated. If the new solution is nondominated and can be added to P_E, then it is also added to P_P, removing all other solutions in the set that it dominates.

A.3 Method Modifications for Dynamic Problem

The four comparison algorithms were all designed for problems with one decision variable type. However, the dynamic benchmark uses both binary and continuous variables. Therefore, modifications were needed to some of the algorithmic functions.

The first modification was made during the initialization of the algorithm. In addition to the packing chromosomes, random t_{fill} arrays are generated. Figure A.5 shows a schematic of the algorithm developed to generate these arrays. The time to fill the first box is randomly selected from a reasonable timeframe. In this case, the range is between 700 seconds, shortly after the first batch of cookies is removed from the oven, and 1200 seconds, when the next batch is removed. The next step is to determine the number of bins to open in t_{fill}. This value is selected randomly either from the set of all possible values or from a set of values chosen with expert guidance. For this problem, the expert set includes values such that each box would

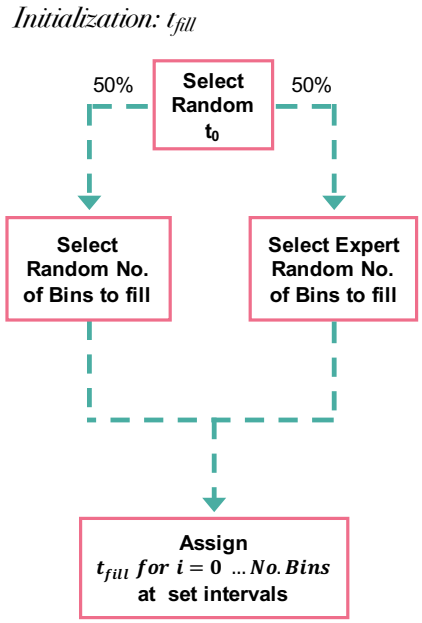


Figure A.5: Initialization scheme for the t_{fill} array in the dynamic benchmark.

be at least 50% filled if a new one is not opened. The t_{fill} array is set at evenly spaced intervals between the first fill time and 600 seconds after the last batch exits the oven.

The next modification adapted the bin packing heuristic discussed in [103] to the dynamic framework. The first change was made to the residual matrix. Originally, the residual matrix keeps track of how much weight and height could be added to each open bin. In the dynamic problem, the physicality constraints of concern are the number of cookies in each box and the difference in time between removal from the oven and placement in the box.

A corresponding change was made in the *Packable* Boolean variable, which determines if an item satisfies the capacity constraints of a bin using the residual matrix [103]. Under the dynamic benchmark, the *Packable* variable signifies if a

cookie both meets a bin's capacity constraint and is taken out of the oven before the bin is set to be filled. Therefore, when moving an item, the residual values for each bin are updated using Eq. A.1 in the dynamic benchmark.

$$\begin{aligned} r_{i,1} &= r_{i,1} - 1 \\ r_{i,2} &= t_{fill,i} - (b_k)_j \Delta t_{batch} \end{aligned} \tag{A.1}$$

The first residual value $r_{i,1}$ relates to the available capacity in a box. When a new box is opened, this value is initialized at the box capacity, and each time a cookie is added, its residual capacity decreases by one. The second residual value $r_{i,2}$ is related to the availability of the cookie at the time box i is filled. Both residual values need to be nonnegative for *Packable* to be true.

The dot product strategy was also changed. In the original heuristic, the weighted dot product given in Eq. A.2 is calculated for every open bin, and the bin that maximizes this value is chosen to store item j [103].

$$w_1 \cdot c_j \cdot r_{i,1} + w_2 \cdot h_j \cdot r_{i,2} \tag{A.2}$$

In Eq. A.2, w_1 and w_2 represent weights used to normalize the dot product in each dimension, which are set as the average weight and height of the items to be sorted. The variables c_j and h_j represent the weight and height of an individual object. To adapt this to the dynamic benchmark, the dot product becomes Eq. A.3.

$$w_1 \cdot r_{i,1} + w_2 \cdot T(t_{fill,i}) \cdot r_{i,2} \tag{A.3}$$

To maintain the intent of the strategy, each of the dot products were related to the physicality constraints. The first product focuses on the available capacity in a box.

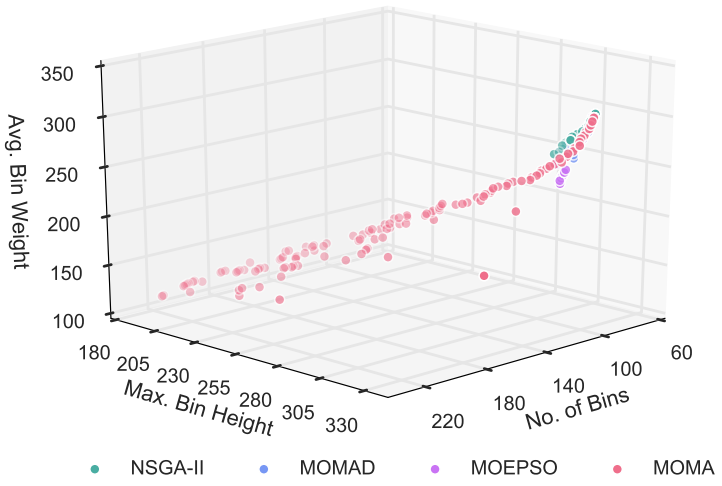


Figure A.6: Scatter plot of approximate sets found for Experiment 8 of the static benchmark.

Therefore, w_1 was set at 1, and c_j was removed in the dynamic benchmark. The second product focuses on the availability of a cookie to move into box i. The chances of a cookie moving into a box are dependent on when it's removed from the oven and how hot it is, so w_2 was set to be the average temperature of a cookie during cooling, and the height was replaced with the temperature of the cookie when box i is filled.

A.4 Static Benchmark Results

To gain insight into MOEAs, the four comparison algorithms were evaluated on their performance in the static bin packing problem described in Section A.1.1. 2DCPackGen was used to generate twenty random instances of the general problem to evaluate the algorithms under a variety of conditions [160]. This section describes which algorithms performed better and discusses initial findings that were used to

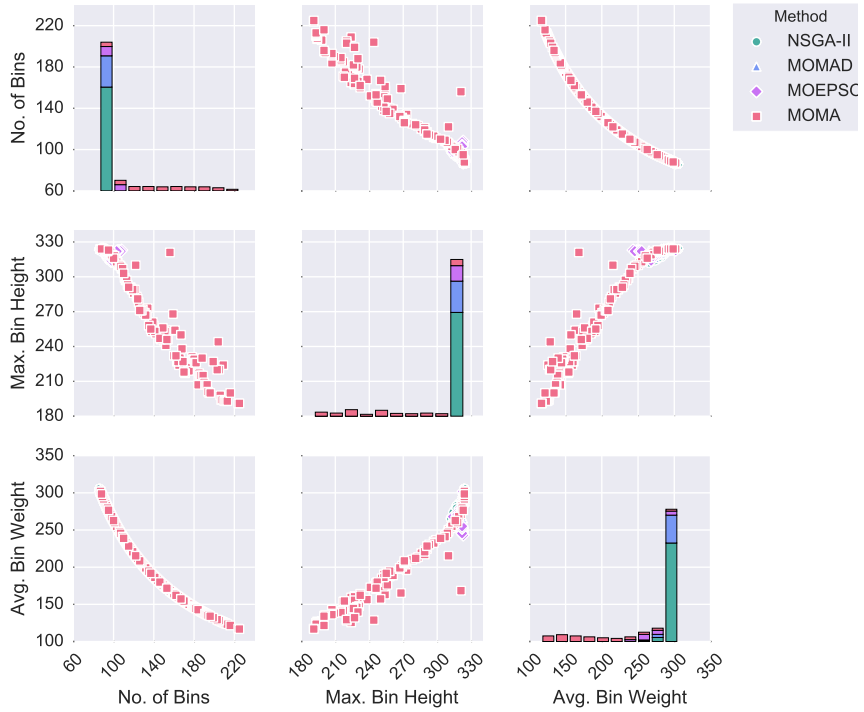


Figure A.7: Scatter matrix of approximate sets found for Experiment 8 of the static benchmark.

develop GAMMA-PC in Appendix B.

Figures A.6 and A.7 present the objective space for one of the scenarios. The results presented here are characteristic of many of the experiments in the static benchmark. Figure A.6 is a 3D scatterplot of the objective space and shows that the approximation set for MOMA spanned much more of the space than the other three algorithms. The approximation sets for the other three clustered in a region of the objective space with a low number of bins. This observation is reinforced by Fig. A.7 as the approximation sets for NSGA-II and MOMAD are contained entirely in the lowest bar for the number of bins. Consequently, the maximum bin height and average bin weights achieved by these approximation sets were concentrated on the higher end of the spectrum. The approximation set for MOEPSO showed a little

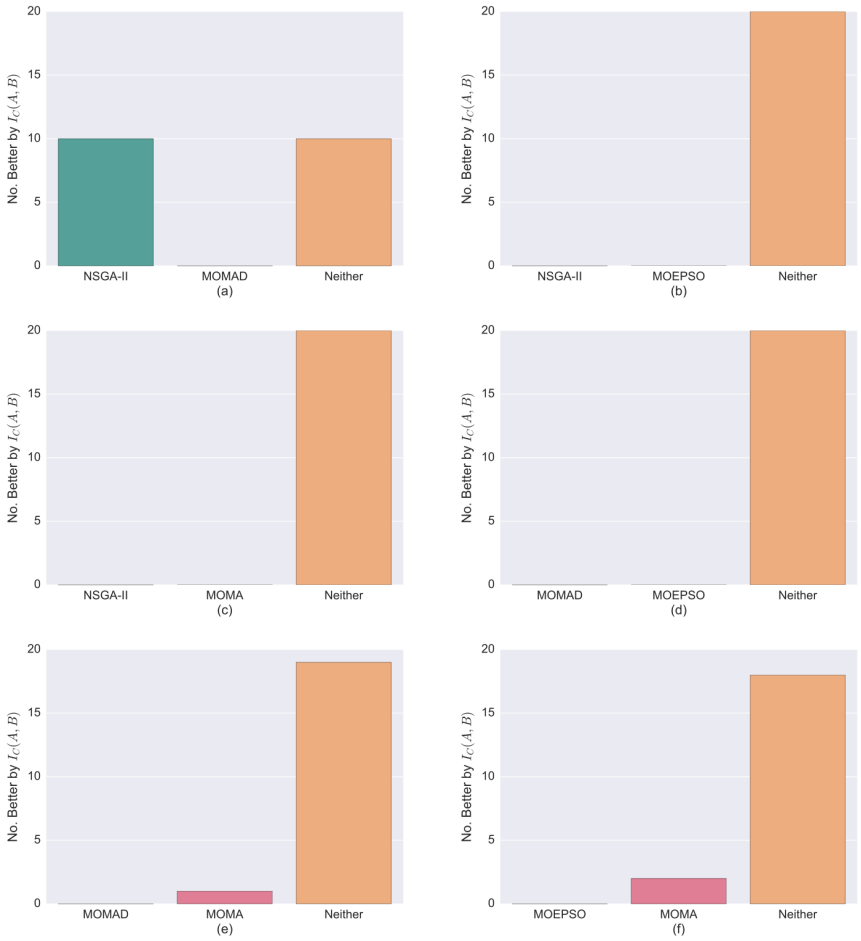


Figure A.8: Binary coverage interpretation results for the twenty static benchmark experiments.

more variation but not as much as MOMA's. The approximation set for MOMA exhibited the expected distribution between the average bin weight and the number of bins, since these objectives are indirectly related.

Figures A.8 and A.9 presents of the binary interpretation function for $I_C(A, B)$ and $I_\epsilon(A, B)$ respectfully. (For more information about these functions, refer to Appendix B.4.3.) The most striking feature of these graphs is that for almost all of the experiments, none of the algorithms could be proven to be better than another. For

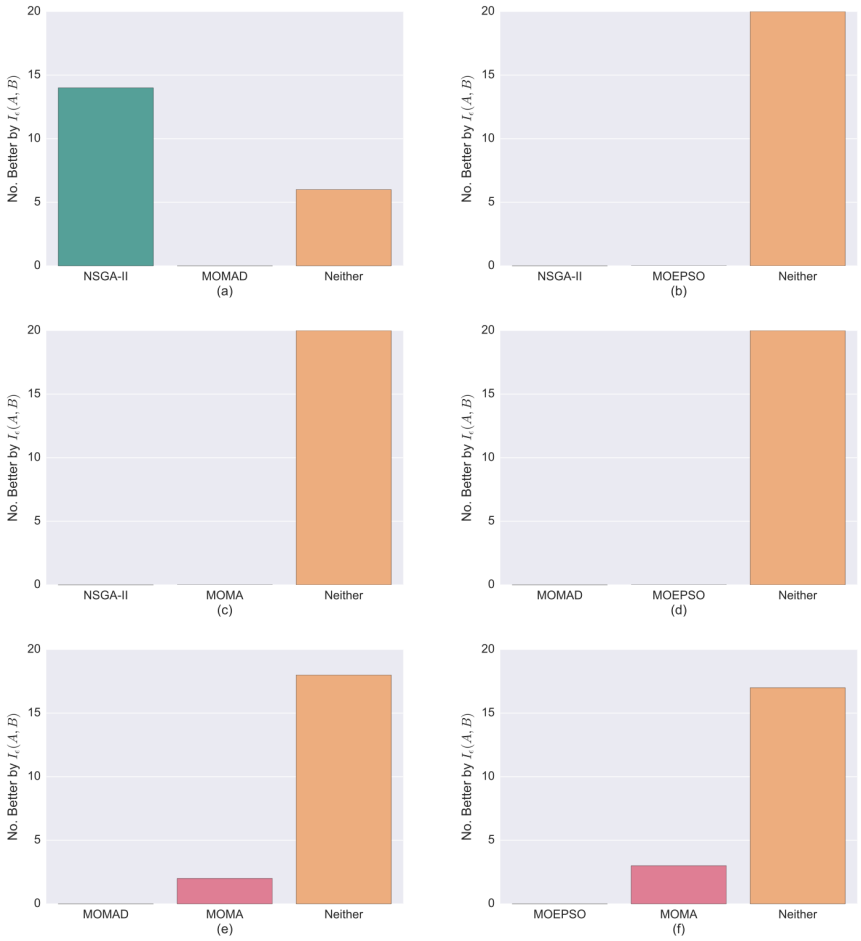


Figure A.9: Binary-ϵ indicator interpretation results for the twenty static benchmark experiments.

the comparisons between NSGA-II and MOMAD, the approximation sets produced by NSGA-II were better in roughly half of the experiments. MOMA also produced a few approximation sets that proved to be better than MOMAD's or MOEPSO's. However, with the large number of comparisons disproving $A \triangleright B$, it would be difficult to choose the appropriate scheme based on the static benchmark alone. Either the static bin packing problem presented little difficulty, or the algorithms produce similar levels of performance.

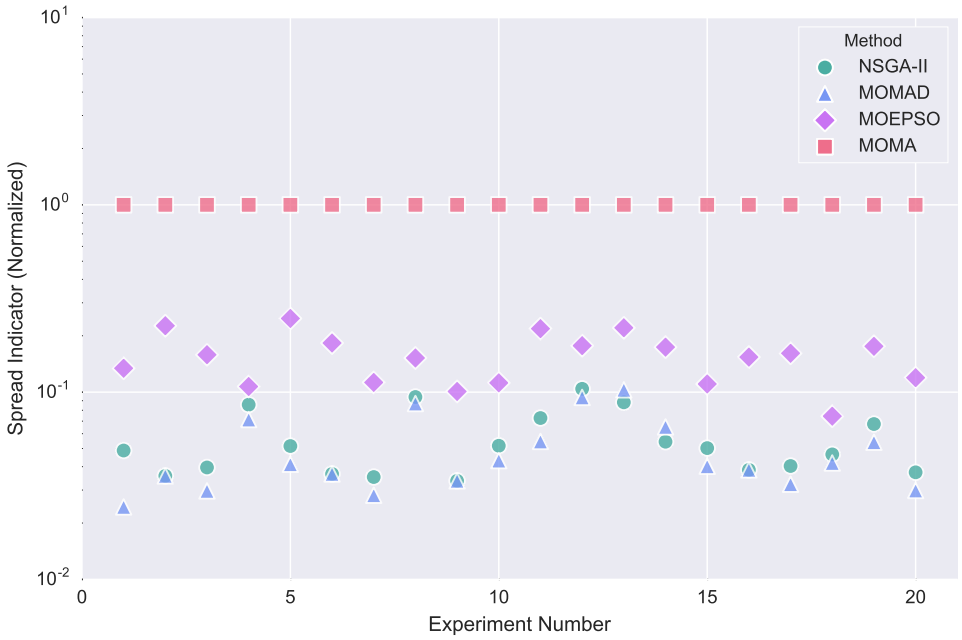


Figure A.10: Normalized spread indicators for static benchmark experiments on a log-scale. The indicators were normalized based on the largest spread in each experiment.

When considering the diversity of the approximation sets, MOMA exhibited the largest range. Figure A.10 presents the spread indicator for the approximation sets in each of the static benchmark experiments. The indicators were normalized based on the largest spread value, which was found to be MOMA's in every experiment. The spread of the approximation sets for MOMA were large enough that Fig. A.10 uses a log scale to show the differences between the other three algorithms. The approximation sets for NSGA-II and MOMAD exhibited similar levels of diversity, while those for MOEPSO were somewhat more diverse.

Based on the performance exhibited in the static benchmark problem, NSGA-II and MOMA were selected to be used in the dynamic benchmark. The approximation sets produced by NSGA-II proved to be better than MOMAD's in approximately half

of the experiments, and both exhibited similar levels of solution diversity. NSGA-II could not be proven to be better than MOEPSO or MOMA, though. Since MOMA produced a few better sets than MOEPSO and had much larger diversity in its approximation sets, it was selected over MOEPSO.

The biggest lesson from these results was that problem-specific knowledge would be important for optimization in the dynamic benchmark. Both MOMA and MOMAD included local search components, but the local search used in MOMA was based on the variable length representation of the loading, whereas MOMAD used the chromosome representation. For a bin packing problem, the information about items in one bin is more useful than where the item is located in the chromosome. Therefore, the diversity achieved using MOMA was much greater. Also, given the number of subproblems MOMAD decomposed the bin packing problem into, there was not a good balance between exploration and exploitation in MOMAD. These lessons were used to develop GAMMA-PC in the dynamic benchmark discussed in Appendix B.

APPENDIX B

A GREEDY MEMETIC ALGORITHM FOR A MULTIOBJECTIVE DYNAMIC BIN PACKING PROBLEM FOR STORING COOLING OBJECTS

The following text was submitted to *Journal of Heuristics* in support of the work completed in this dissertation. It presents a mathematical treatment of GAMMA-PC and compares the performance of the method to other state-of-the-art algorithms.

B.1 Abstract

In this paper, a multiobjective dynamic bin packing problem for storing cooling objects is introduced along with a metaheuristic designed to work well in its mixed-variable environment. The dynamic bin packing problem is based on the idea of cookie production at a bakery, where cookies arrive in batches at a cooling rack with limited capacity and are packed into boxes with the three competing goals. The first goal is to minimize the number of boxes used. The second objective is to minimize the initial heat of each box, and the third is to minimize the maximum time until the boxes can be moved to the storefront. The metaheuristic developed here incorporated greedy heuristics into an adaptive evolutionary framework with partial decomposition into clusters of solutions for the crossover operator. The new metaheuristic was applied to a variety benchmark bin packing problems and to a small and large version of the dynamic bin packing problem. It performed as well as other metaheuristics in the benchmark problems and produced more diverse solutions in the dynamic problems. It performed better overall in the small dynamic problem, but its performance could not be proven to be better or worse in the large dynamic problem. **keywords:** Dynamic bin packing problem, Multiobjective combinatorial

optimization, Metaheuristics

B.2 Introduction

The bin packing problem is a NP-hard combinatorial problem relevant to many real-world systems [161]. Its primary goal is to fit a number of items into as few bins as possible within given constraints. The general bin packing problem has the flexibility to be abstracted to represent ideas such as cloud computing or to be formulated to directly resolve physical space limitations, such as packing boxes for transport.

In the past decade, more attention has been given to the multiobjective needs of the systems that these problems represent. The first multiobjective bin packing problem was extended to minimize both the number of bins and the average deviation from the center of gravity in the bins [130]. Since then, many other studies have included this "load-balancing" objective [162] as well as minimizing the maximum length of a bin [103] and minimizing costs [163].

Traditionally, bin packing problems have been solved using approximate algorithms, such as the *First-Fit* and *Best-Fit* methods [132]. While exact algorithms are available for very small problems, approximate algorithms can produce near-optimal solutions in $O(n \log n)$ time, an advantage for medium to large packing problems. However, when more than one objective is present, approximate algorithms work best when combined with a method suited for the multidimensional objective space.

Bin packing problems with multiple competing objectives are structured according to the standard multiobjective problem format, given in (B.1).

$$\begin{cases} minimize & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_m(\mathbf{x}))^T \\ s.t. & \mathbf{x} \in \Omega \end{cases} \tag{B.1}$$

In (B.1), the decision vector $\mathbf{x}$ belongs to the feasible region Ω. The objective vector $\mathbf{F}$ translates $\mathbf{x}$ into the objective space through m objective functions. The decision vector $\mathbf{x}$ for a bin packing problem with N items is typically represented by two binary matrices such as those given in (B.2) and (B.3).

$$x_{ij} \in \{0,1\},\ \forall\, i \in \{1,...,\overline{M}\},\ j \in \{1,...,N\} \tag{B.2}$$

$$y_i \in \{0,1\},\ \forall\, i \in \{1,...,\overline{M}\} \tag{B.3}$$

Here, x represents the loading matrix, and y is the open bins vector. The index i is the counter for the bins, and j is the counter for the objects. The variable $\overline{M}$ represents the theoretical maximum number of bins, so x is an $N \times N$ matrix, and y is an array of length N.

The best solutions in Ω involve trade-offs among the competing objectives. Together, these solutions are called the **Pareto set**, and their image in the objective space is the **Pareto front**. A feasible solution is proven to belong to the Pareto set by showing that it is not *dominated* by any other solution, defined formally below [128].

Definition B.1. *In a multiobjective problem with m objectives, an objective vector $u = (u_1,...,u_m)^T$ **dominates** another vector $v = (v_1,...,v_m)^T$ iff $\forall\theta \in \{1,...,m\}$, $u_\theta \leq v_\theta$ and $u \neq v$, written $u \prec v$.*

An algorithm may not reach the true Pareto front during its finite computation, so an instance of a set of nondominated feasible solutions is called an **approximation set** [129].

Multiobjective evolutionary algorithms (MOEAs) have been the most popular method for solving multiobjective optimization problems given their ability to quickly find approximations to the Pareto front [128]. Consequently, for multiobjective bin

backing problems, the heuristics of approximate algorithms have been combined with MOEAs, such as the multiobjective evolutionary particle swarm optimization (MOEPSO) [130] or the nondominated sorting genetic algorithm II (NSGA-II) [103]. However, as an extension of evolutionary algorithms, MOEAs suffer from a weak exploitation ability and can be slow to converge. Using an ensemble approach can greatly improve this limitation. Recent studies have found better speed and accuracy with **memetic** algorithms [128], the ensemble of MOEAs and local search methods.

This study makes two main contributions. To the best of our knowledge, this paper is the first to formulate a dynamic bin packing problem in this manner. It is different from previous dynamic bin packing problems where items have an arrival time and a departure time, such as those used to model cloud computing [131]. The dynamic bin packing problem introduced here does not include a departure time. Also, the items have time-varying characteristics, and an intermediate "holding" area is available so that the arrival time does not necessarily coincide with when the item is placed in a bin. Consequently, the bin packing decision vector includes a continuous variable representing the time each bin is filled.

The second contribution is a metaheuristic algorithm, named the GRASP-enabled adaptive multiobjective memetic algorithm with partial clustering (GAMMA-PC), which is capable of performing in this mixed-variable environment. In this study, GAMMA-PC was applied to standard bin packing problems as a basis for wider comparison and then was applied to both a small version and a large version of the dynamic bin packing problem introduced here. It will be shown that GAMMA-PC performs as well as other state-of-the-art MOEAs on standard problems. It is also better at exploring the objective space in a mixed-variable environment. However, while it performs better overall in the small version of the dynamic problem, it will be demonstrated that the right balance between exploitation and exploration has

not been found yet, degrading its performance in the large dynamic problem.

This paper describes the algorithmic flow of GAMMA-PC and its features in Sec. B.3. Section B.4 introduces the dynamic bin packing problem and the methodology to evaluate the performance of the algorithm. Section B.5 compares the performance of GAMMA-PC to other state-of-the-art metaheuristics, and Sec. B.6 draws conclusions.

B.3 Algorithm

GAMMA-PC combines approaches that have shown promising performance for specific aspects of the dynamic bin packing problem. The general framework is based on NSGA-II [102] combined with local search techniques. It has previously been shown that embedding packing heurisitics in a greedy randomized adaptive search procedure (GRASP) performs better than using approximate algorithms alone, especially for problems with special packing requirements [143]. Therefore, to handle the additional complexity of the time requirements, GRASP is used to perform the packing and to govern the local search procedures.

Throughout the calculations, three populations are maintained: P, Q, and P_{EA}. The solutions in P are the parent solutions selected to be used in the genetic operators, and the solutions in Q are those found by the genetic operators and the local search procedures. The set P_{EA} is the external archive of nondominated solutions and is updated every generation. The general outline of GAMMA-PC is as follows.

1. Initialize set Q using *ConstructDynamicBPP*

2. While the stopping condition is not satisfied, do:

3. Select set P using NSGA-II-styled binary selection,

4. Update P_{EA},

Table B.1: GRASP procedure for creating a new solution.

	Algorithm: GRASP-DBPP
	Input: $j = \{1, ...N\}$,
	optional: chromosome representation
	Ouput: Solution
1	**if** chromosome representation:
2	**x** := Construct-DBPP-Chrom(input)
3	**else**:
4	**x** := Construct-DBPP-New(input)
5	**end if**
6	Create Solution from **x** and evaluate **F**
7	Find a Neighbor using LocalSearch(Solution)
8	**return** BestSolution(Solution, Neighbor).

5. Update Q using *Crossover-PC*,

6. Update Q by *Mixed-Variable-Genetic-Mutation*,

7. Transform new solutions in Q using *ConstructDynamicBPP*,

8. Add solutions to Q using *Adaptive-Local-Search*,

9. Every g generations, *Truncate-External-Archive*,

10. *Update-Operator-Probabilities*,

11. End while, and return P_{EA}.

In Step 1 of GAMMA-PC, Q is initialized using the algorithm *GRASP-DBPP*, shown in Table B.1, to construct enough solutions to fill the initial parent population. The *GRASP-DBPP* sequence can be called with a real-valued chromosome representation of the items, but during the initialization step, all new solutions are constructed from scratch. The decision vector **x** is constructed according to the algorithm shown in Table B.2. With *Construct-DBPP-New*, items are assigned to bins sequentially based on a randomly chosen packing heuristic θ_i from a set of m

Table B.2: GRASP heuristic to construct new solution.

	Algorithm: Construct-DBPP-New
	Input: $j = \{1, ...N\}$
	Ouput: $vlrep$, x, y, t_{fill}
1	t_{fill} := $Nx1$ array with $t_{fill,i} = 0 \; \forall \; i \in \{1, ..., N\}$
2	y := $Nx1$ array with $y_i = 0 \; \forall \; i \in \{1, ..., N\}$
3	RCL_t = InitializeGreedyFunction()
4	Select $t_{fill,1}$ at random from RCL_t
5	$vlrep$:= [[1]] and set $y_1 = 1$
6	Select θ_i at random from range(m)
7	**for** $j \in \{2, ..., N\}$ **do**
8	Construct RCL_i based on mode θ_i
9	Select i at random from RCL_i
10	**if** $y_i = 0$:
11	Append [j] to $vlrep$
12	Select θ_t at random from range($m - 1$)
13	Construct RCL_t based on mode θ_t
14	Select $t_{fill,i}$ at random from RCL_t
15	AdaptGreedyFunction(t_{fill})
16	$y_i = 1$
17	**else**:
18	Append j to $vlrep_i$
19	AdaptGreedyFunction($vlrep$)
20	**end if**
21	**end for**
22	Construct x based on $vlrep$

options, each tailored to minimize an individual objective function. The selected heuristic returns a restricted candidate list of bin options using a **cardinality** restriction, where the list only contains the best β candidates [144]. Whenever a new bin is opened, a $t_{fill,i}$ value is chosen from a special continuous greedy function that will be discussed later. The bin packing is stored in a variable length representation during the procedure, which is translated into the x loading matrix after all the items have been packed.

After creating a new solution, *GRASP-DBPP* finds one neighbor using a randomly selected local search operator. GAMMA-PC uses $m+1$ local search operators, where each operator explores a carefully selected local neighborhood to minimize one

of the objective functions. Two operators represent the first objective of bin packing: minimizing the number of bins. One search operator moves a solution toward having fewer bins, while the other repacks bins to find new nondominated solutions. The latter is chosen if a solution has already reached the theoretical minimum number of bins. After finding a neighbor, the dominant solution is returned.

The parent population P is selected every generation using NSGA-II-styled binary selection. Therefore, sets P and Q are combined and sent through the *fast-non-dominated-sort* procedure described by [102]. Then, the best solutions are collected based on nondomination levels and crowding distance assignments and sent to a binary tournament called the *Crowded-Comparison Operator* to select a new set P. More details about these procedures can be found in [102]. The solutions in P are then used in the genetic operations of *Crossover-PC* and mutation.

The genetic operations in Steps 5 and 6 are modified from their traditional form to account for the mixed-variable environment. Both genetic operators handle the chromosome representation of the packing and the t_{fill} decision variable separately. The crossover operator performs a single-point crossover on the chromosome representation and then on t_{fill} in its matrix form. The modification to the genetic mutation operator is more complicated. The mutation first performs a two-point swap in the chromosome representation as it would before mixed-variables were introduced. Then, the $\mathbf{t}_{fill}$ array is modified using a new technique for a bin packing problem.

Previous research with mixed-variable optimization has advised the mutation of a continuous variable using a normal distribution based on the variability present in the population members [145]. The function to perform this mutation, translated

Table B.3: Algorithm to produce standard deviations for mutation of $\mathbf{t}_{fill}$ matrices.

	Algorithm: Bin-Packing-Time-Sequencing
	Input: $S = \{t_{fill,i} \neq 0.0,\ \forall\ i \in \{1, ..., \overline{M}\}$, $\mathbf{t}_{fill} \in \{\mathbf{t}^1_{fill}, \mathbf{t}^2_{fill}, ..., \mathbf{t}^{pop}_{fill}\}_g\}$
	Ouput: T_{sd}
1	$t_{min} = t_{low} = \Delta t_{batch}$
2	t_{max} = maximum time in S
3	n_{cat} = (t_{max} - t_{min}) / Δt_{batch}
5	$t_{high} = t_{low} + \Delta t_{batch}$
6	$T_{sd} := \varnothing$
7	**for** *cat* in range(n_{cat}) **do**
8	$t_{cat} = \{t \in S : t_{low} \leq t < t_{high}\}$
9	$\sigma_{cat} \leftarrow$ Calculate the standard deviation of t_{cat}
10	Add (t_{low}, t_{high}, σ_{cat}) to T_{sd}
11	$t_{low} = t_{high}$
12	t_{high} += Δt_{batch}
13	**end**

into the particulars of the dynamic problem, is shown in (B.4).

$$t_{fill,i,g+1} = t_{fill,i,g} + \mathcal{N}(0, \sigma_{i,g}),\ \forall\ i \in \{1, ..., \overline{M}\} \tag{B.4}$$

In (B.4), g represents the generation, and $\sigma_{i,g}$ is the standard deviation of the values for $t_{fill,i}$. Because the dynamic bin packing problem uses a variable number of bins and the bins are not necessarily sorted by their fill time, this can lead to errors. To ensure that the mutation produces a usable $\mathbf{t}_{fill}$ matrix, the function suggested by [145] is modified to become (B.5).

$$t_{fill,i,g+1} = t_{fill,i,g} + \mathcal{N}(0, \sigma_{cat}),\ \forall\ i \in \{1, ..., \overline{M}\} \tag{B.5}$$

Here, the normal distribution is based on the standard deviation of a category of time values. To find these values, all of the fill times used by the solutions in set P are gathered together and separated into categories based on the *Bin-Packing-*

Table B.4: GRASP heuristic to decode a chromosome representation into a new solution.

	Algorithm: Construct-DBPP-Chrom
	Input: *chrom* (chromosome representation of packing),
	$t_{fill,s}$ (suggested by genetic operations)
	Ouput: *vlrep*, x, y, t_{fill}
1	$t_{fill} := Nx1$ array with $t_{fill,i} = 0 \; \forall \; i \in \{1, ..., N\}$
2	$y := Nx1$ array with $y_i = 0 \; \forall \; i \in \{1, ..., N\}$
3	RCL_t = InitializeGreedyFunction()
4	Select $t_{fill,1}$ from $t_{fill,s}$ or at random from RCL_t
5	*vlrep*, *chrom* = InitializeFirstBin(*chrom*, t_{fill})
6	$y_i = 1$
7	Select θ_i at random from range(m)
8	**for** j **in** *chrom* **do**
9	Construct RCL_i based on mode θ_i
10	Select i at random from RCL_i
11	**if** $y_i = 0$:
12	Append [j] to *vlrep*
13	Select θ_t at random from range($m - 1$)
14	Construct RCL_t based on mode θ_t
15	Select $t_{fill,i}$ from $t_{fill,s}$ or at random from RCL_t
16	AdaptGreedyFunction(t_{fill})
17	$y_i = 1$
18	**else**:
19	Append j to $vlrep_i$
20	AdaptGreedyFunction(*vlrep*)
21	**end if**
22	**end for**
23	Construct x based on *vlrep*

Time-Sequencing algorithm, shown in Table B.3. This algorithm returns a list of standard deviations along with the temporal boundaries for each category so that when mutation occurs, each $t_{fill,i}$ can be connected with the standard deviation in its time category.

In Step 7, GAMMA-PC translates the chromosomes and t_{fill} arrays into new solutions. *GRASP-DBPP* sends the chromosome representation and t_{fill} array to the *Construct-DBPP-Chrom* algorithm shown in Table B.4, which performs the decoding in a similar manner to *Construct-DBPP-New*. One difference is the order in which

the items are considered. Each number in the chromosome corresponds to an item to be packed, and the order listed in the chromosome determines the sequence of items packed. Another difference is that the algorithm tries use t_{fill} values from the array produced by the genetic operations before finding a new feasible time value for a bin. When initializing the first bin, each item is considered in sequence until one is found that can be placed in that bin without violating time constraints. The index number for this item is removed from the list, and then each item is packed in order of the newly shortened chromosome.

In Step 8 of GAMMA-PC, local search is performed on the new solutions in Q with probabilities that are updated every generation. The same $m + 1$ local search operations used in *GRASP-DBPP* are called here, searching for N_{LS} neighbors instead of only one. On even generation numbers, the solutions in P_{EA} are clustered into groups based on the number of bins used, and a random solution in each cluster is sent to a randomly chosen local search operator. The solutions at the extremes are always included, searching near m solutions in P_{EA}, where each solution has a fitness value representing the minimum found for the corresponding objective function. Finally, every four generations, local search is performed across P_{EA} according to the same probabilities used for the local search of Q. Any neighbor found to be nondominated to the input solution is added to Q. After every g generations, P_{EA} is truncated based on the crowded distance assignment value to keep the size of the archive below a preset level.

B.3.1 Continuous Greedy Function for Selecting Fill Times

The continuous greedy function to select bin fill time values combines the idea of Monte Carlo selection with that of a restricted candidate list. The function RCL_t maintains the timeline and keeps track of the available space in the intermediate

Table B.5: Algorithm to initialize t_{fill} restricted candidate list.

	Algorithm: InitializeGreedyFunction
	Input: R, N, n_{batch}, Δt_{batch}
	optional: vlrep, t_{fill}
	Ouput: RCL_t
1	$b_n = N$ / n_{batch}
2	$t_{range} = \{b\Delta t_{batch}\}\ for\ b \in \{1, ...b_n + 1\}$
3	$space = \{R - bn_{batch}\}\ for\ b \in \{1, ...b_n\}$
4	Append $(R - b_n n_{batch})$ to *space*
5	**if** *optional* arguments given:
6	**for** every bin i in *vlrep* **do**
7	AdaptGreedyFunction(*vlrep*[i], $t_{fill,i}$)

holding area as a function of time. When the function is called upon to generate a new fill time, it restricts the timeline and returns a randomly selected time from within that range.

Table B.5 presents the algorithm used to initialize RCL_t. The timeline begins as a list of arrival times for each batch of items with an additional period at the end. The list generator shown for t_{range} is based on the assumption of evenly spaced arrival periods and can be easily modified for other regimes. The list generator for *space* assumes standard group sizes moving into the holding area with capacity R, with the last value added to *space* steady state with the previous. If RCL_t is initialized from an existing solution, the variable length representation and the t_{fill} array would be used to adapt the greedy function, increasing the space values to match the removal of items from the holding area at given fill times.

When RCL_t is used to generate a new fill time, the space and timeline are converted into a probability density function. The *space* list in RCL_t is allowed to hold negative values, but these are converted to 0% probability in the probability density function. The probability density function is transformed into a cumulative density function, which is used to find a new time value.

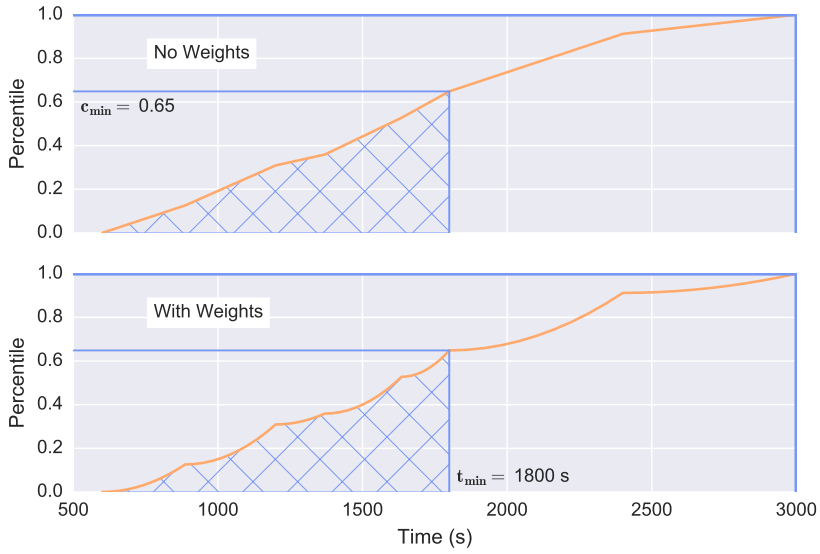


Figure B.1: Illustration of the continuous greedy function, with and without weights. A new time value is chosen by randomly selecting a percentile within the open region defined by c_{min} and c_{max} and inserting it into the inverse cumulative density function.

Fig. B.1 presents two different versions of the cumulative density function for a bin packing case with a timeline from 600 to 3000 seconds. The timeline is first restricted by setting a minimum value at the arrival time of the item under consideration. In this case, the maximum time value is set at the end of the timeline, but if the holding area were ever filled beyond capacity, the time when the overflow occured would be set as the maximum time. Corresponding percentiles are found for the minimum and maximum time values, and a random percentile is chosen within that range. The chosen percentile is then converted back into a new time value using an inverse cumulative density function.

There are two different continuous greedy functions shown in Fig. B.1 to emphasize that the new time generator for RCL_t can be adjusted to the problem at hand. The top figure shows the cumulative density function without modification, so an

Table B.6: Crossover operator with partial clustering.

	Algorithm: Crossover-PC
	Input: P, f_{ran}
	Ouput: Q
1	Split P into P_{ran} and P_c based on f_{ran}
2	$Q_{ran} \leftarrow$ Mixed-Variable-Genetic-Crossover(P_{ran})
3	Update ideal values z^* using P_{EA}
4	$N_c = 2m$
5	Sort solutions from P_c into N_c Tchebycheff clusters
6	$Q_c := \emptyset$
7	**for** c in range(N_c) **do**
8	**if** cluster c is not empty:
9	New solutions $\leftarrow$ Mixed-Variable-Genetic-Crossover(cluster c)
10	Add New solutions to Q_c
11	**else:**
12	Update-Mutation-Rate
13	**end if**
14	**end for**
15	$Q = Q_{ran} + Q_c$

item has the same probability of being chosen within any given period between t_{range} values. In the bottom figure, the probability is modified so that the total probability in the period remains the same, but the probability of chosing a time value increases linearly within that period. Therefore, the top and bottom continuous greedy functions hit the same percentile values at the times listed in t_{range}, but the shapes of the functions between those points are different.

B.3.2 Crossover with Partial Clustering

The standard genetic crossover used in NSGA-II is modified in GAMMA-PC to include partial clustering, as shown in Table B.6, with the goal of balancing exploration and exploitation of the objective space. The *Crossover-PC* operator first sends a fraction f_{ran} of P straight to the crossover operator to be randomly paired and mixed. Then, it carefully sorts the remaining solutions into N_c clusters before sending each cluster to the crossover operator. All of the solutions produced

by the multiple crossovers are combined to make up Q.

Crossover-PC performs the grouping of the clusters using the Tchebycheff approach [146]. To support the use of this approach without knowledge of the Pareto front, the ideal values z^* are updated every generation to reflect the most ideal points found in the objective space for every objective θ. The $m \times 1$ array z^* is set using (B.6).

$$z^*_\theta = \min_{\mathbf{x} \in \Omega} f_\theta(\mathbf{x}) \tag{B.6}$$

To perform the sorting, N_c random weight vectors of the form $\lambda = (\lambda_1, ..., \lambda_m)^T$ are produced such that $\sum_{\theta=1}^{m} \lambda_\theta = 1$ and $\lambda_\theta \geq 0$ for all $\theta = 1, ..., m$ [146]. Then, the objective vectors for each solution in P_c are transformed into N_c single objective vectors through (B.7) with the random weight vectors.

$$g^{te}(\mathbf{x}|\lambda) = \max_{1 \leq \theta \leq m} \left\{ \frac{\lambda_\theta}{w_\theta} (f_\theta(\mathbf{x}) - z^*_\theta) \right\} \tag{B.7}$$

This procedure is similar to the process of decomposition in MOEA/D-type algorithms but is only applied during the crossover operation. The form of (B.7) is also modified from the Tchebycheff approach generally used by MOEA/D [146] to reflect the purpose of the weight in the Tchebycheff norm, which is to normalize various criteria [147]. The λ weight vector is still included in (B.7), but it is divided by the vector w. This is calculated in similar manner as z^* in (B.6), except finding the maximum value present for each objective among the solutions in P_{EA}. The inclusion of w ensures that the clusters are formed throughout the objective space, even if one objective function explores a much larger range than the others. After the single objective fitness values are calculated for a solution, it is placed in the cluster with the weight vector resulting in the smallest single objective value.

Table B.7: Algorithm to update local search probabilites.

	Algorithm: Update-Operator-Probabilities
	Input: P_{EA}, p_{ls}^{g}, $p_{ls,\theta}^{g}$ for θ = 1,...,$m-1$
	Ouput: p_{ls}^{g+1}, $p_{ls,\theta}^{g+1}$ for θ = 1,...,$m-1$
1	$IdealNSize := m \times 1$ array of zeros
2	**for** θ in range(m) **do**
3	$\quad IdealNeighbors = \{u \in P_{EA} : u_\theta - z_\theta^* \leq 0.10 z_\theta^*\}$
4	$\quad IdealNSize_\theta = \lvert IdealNeighbors \rvert$
5	**end for**
6	$p_{ls}^{g+1} \leftarrow$ Update-Local-Search-Rate(p_{ls}^{g}, $IdealNSize$)
7	$a = \min(IdealNSize)$
8	$b = \max(IdealNSize)$
9	$p_{move} = \left(1 - a/b\right) p_{ls,\theta(b)}^{g}$
10	$p_{ls,\theta(b)}^{g+1} = p_{ls,\theta(b)}^{g} - p_{move}$
11	$p_{ls,\theta}^{g+1} \leftarrow$ Redistribute p_{move} to other local search operators for $\theta = \{1, ..., m\} \setminus \{\theta(b)\}$
12	Ensure the sum of $p_{ls,\theta}^{g+1}$ is 1.0 for $\theta = 1, ..., m$

B.3.3 Updating Operator Probabilities

The first "A" in GAMMA-PC refers to the adaptation performed every generation to move the operator probabilities toward areas of need during calculations. The first change is made during *Crossover-PC* in Step 5. For every empty cluster found, the mutation rate increases by a small amount. This change occurs because the single objective fitness values represent different areas of the objective space, so if a cluster is empty, the area governed by that weight vector has not been explored well. The increase is made in the mutation rate to encourage more random exploration.

The local search probabilities are also updated every generation to encourage search in one area or another based on the solutions present in P_{EA}. During calculations, m local search probabilities are maintained: the probability of local search overall p_{ls} and the probabilities of objective-specific local search operators $p_{ls,\theta}$ for the first $m-1$ objectives. The probability of local search for the last objective is implicit because the total should sum to 1. The algorithm to update these prob-

abilities is given in Table B.7. First, the number of solutions in P_{EA} in the local neighborhood of each ideal value are counted, found using the relationship shown in Step 3 of *Update-Operator-Probabilities*. The size of each neighborhood is stored in the $m \times 1$ array $IdealNSize$. The overall probability p_{ls} is increased if any value in $IdealNSize$ falls below a preset minimum value or is decreased if all of the values are above a preset maximum.

Next, the probabilities for the individual local search operators are updated based on the variation in $IdealNSize$. The probability of the local search operator associated with the objective with the largest ideal neighborhood is reduced by a fraction determined by the size of the smallest ideal neighborhood. The portion it loses p_{move} is redistributed fairly among the probabilities for the other objectives. For example, if the probabilities at the end of generation g were $p^g_{ls,1} = 0.25$, $p^g_{ls,2} = 0.25$, and $p^g_{ls,3} = 0.50$ and if $IdealNSize$ were found to be $(2, 4, 8)$, then $p_{ls,3}$ would lose $(1-2/8)0.50$ from its share, or $p_{move} = 0.375$. A fair way to redistribute this between the search operators for the first and second objectives would be to give $p_{ls,1}$ a larger chunk of p_{move} since its neighborhood is smaller. Therefore, the probabilities would be updated to $p^{g+1}_{ls,1} = 0.50$, $p^{g+1}_{ls,2} = 0.375$, and $p^{g+1}_{ls,3} = 0.125$ in the next generation.

B.4 Simulation Methodology

This section describes the test problems and the performance metrics used to evaluate the new method.

B.4.1 Static Test Problem

The static test problem is a general bin packing problem sorting idealized objects into larger bins. It is included to demonstrate how GAMMA-PC performs under a familiar packing problem format. Each object has a weight and a height, and the bins are assigned weight and height limits, as illustrated in Fig. B.2. The objective

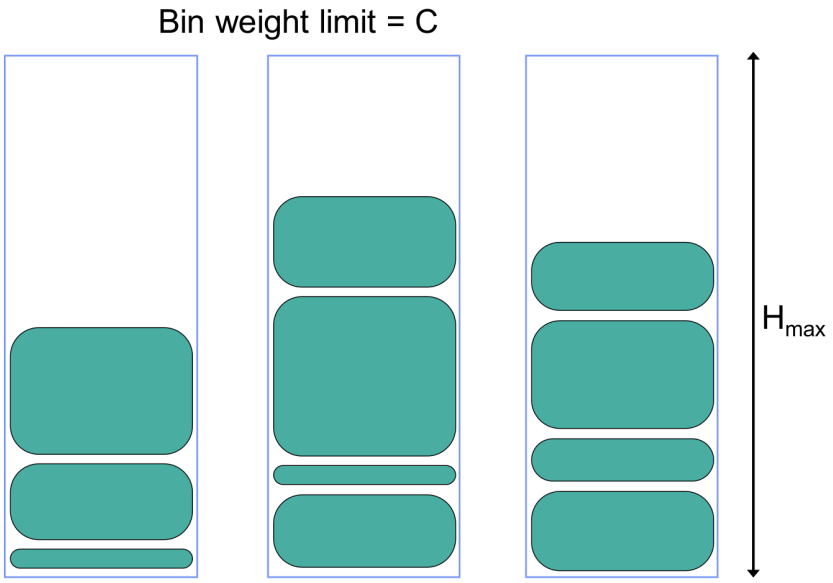


Figure B.2: Illustration of the static benchmark problem setup. Each object has a weight and a height and are sorted into bins with the goals of minimizing the number of bins, the average weight of a bin, and the maximum used height.

of the problem is to minimize the number of bins in use while also minimizing the average weight and the maximum used height of the bins. This is similar to the bin packing problem described in [103].

B.4.1.1 Mathematical Formulation

The mathematical formulation of the static test problem is given from (B.8) to (B.13).

$$min.\ z_1(s) = \sum_{i=1}^{\overline{M}} y_i \tag{B.8}$$

$$min.\ z_2(s) = \frac{\sum_{i=1}^{\overline{M}} \sum_{j=1}^{N} c_j x_{ij}}{\sum_{i=1}^{\overline{M}} y_i} \tag{B.9}$$

$$min.\ z_3(s) = max\left(\sum_{j=1}^{N} h_j x_{ij},\ i \in \{1, ..., \overline{M}\}\right) \tag{B.10}$$

s.t.

$$\sum_{i=1}^{\overline{M}} x_{ij} = 1,\ \forall j \in \{1, ..., N\} \tag{B.11}$$

$$\sum_{j=1}^{N} c_j x_{ij} \leq C y_i,\ \forall i \in \{1, ..., \overline{M}\} \tag{B.12}$$

$$\sum_{j=1}^{N} h_j x_{ij} \leq H y_i,\ \forall i \in \{1, ..., \overline{M}\} \tag{B.13}$$

Objective (B.8) minimizes the number of bins in use, (B.9) minimizes the average weight of a bin, and (B.10) minimizes the maximum used height. The variable c_j represents the weight associated with object x_{ij}, and h_j represents the height. These objectives are bounded by the "no replacement" constraint (B.11), the bin weight capacity constraint (B.12), and the bin maximum height constraint (B.13). The weight and height limits are denoted C and H, respectively.

To generate items for the static problem, 2DCPackGen [160] was initiated using the two-dimensional setting to produce height and weight values for the given number of objects. The objects were made using the Single Bin Size Bin Packing Problem selection. Table B.8 presents the other parameter settings used to generate the problem characteristics. The shapes and sizes were chosen to ensure the benchmark would cover a diverse range of possibilities, and the beta distribution was chosen so that the characteristics would be roughly normal.

B.4.1.2 Statistical Setup

For the static problem, GAMMA-PC was evaluated against four other MOEAs:

- NSGA-II [102],
- multiobjective memetic algorithm (MOMA) [134],
- memetic algorithm based on decomposition (MOMAD) [157],

Table B.8: Parameter selections to generate experimental samples for the static benchmark problem using 2DCPackGen [160].

2DCPackGen Option	Selection
Seed	85518
Number of instances	20
Minimum and maximum size dimension of the large object	300 1000
Minimum and maximum size dimension of the small items	10 100
Characteristic of the size and shape of the large object	Average length and narrow or average length and tall
Characteristic of the size and shape of small items	Small and square, short and tall, long and narrow or big and square
Minimum and maximum number of different item types	500 500
Characteristic of the generator for the number of different item types	Beta distribution w/ $\alpha = 0.5$ and $\beta = 0.5$

- and MOEPSO [130].

These methodologies have previously been applied to combinatorial problems and represent a variety of techniques. NSGA-II was chosen as one of the most popular MOEAs used to validate new algorithms, such as in [158]. MOEPSO was chosen because it was specifically designed for multiobjective bin packing problems, and MOMA was chosen to represent a memetic version of NSGA-II. Finally, MOMAD was chosen as one of the newest memetic algorithms with a flexible framework.

To evaluate GAMMA-PC across of variety of specific instances, twenty cases were generated with 2DCPackGen [160], and the algorithms were used to calculate solutions to each experiment. The basis of the evaluation was 25,000 function evaluations, and the metrics used for the evaluation are described in Section B.4.3.

B.4.2 Dynamic Test Problem

A real world example of a bin packing problem can be seen in the daily operation of a bakery. Many cookies, muffins, or other items are baked and placed into boxes throughout the day. Suppose a baker wanted to optimize their process to something other than boxing the oldest and coldest cookies first. Then, the bin packing problem for the baker would be much more complex than the general problem described in Sect. B.4.1, and the new dynamics would introduce features into the bin packing problem that are relevant to other real world problems, such as resource allocation.

The dynamic test problem established here concentrates on baking a set of n cookies. It is assumed that n is larger than the capacity of the oven, so the cookies bake in batches. Before being placed into bins, or cookie boxes, the cookies must cool off to avoid moisture buildup. However, the cooling racks can only hold a limited amount of cookies, so the baker must start filling the boxes before the end of the cookie baking session. As illustrated in Fig. B.3, each box has a capacity of C cookies,

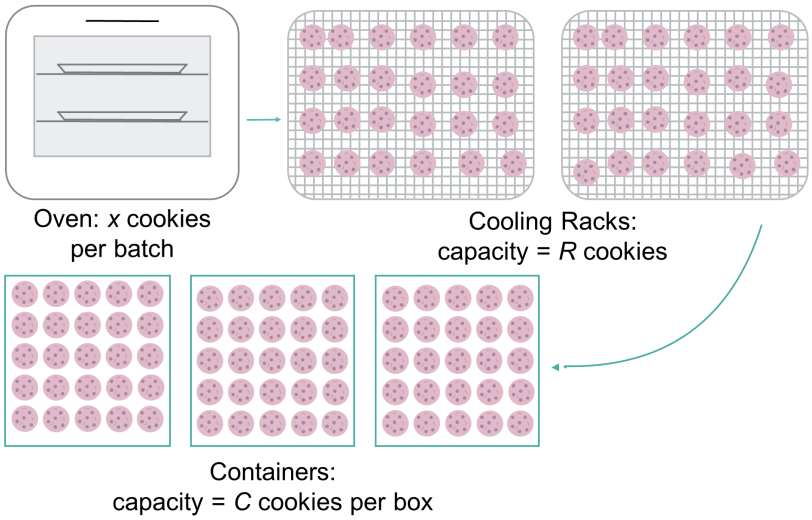


Figure B.3: Illustration of dynamic benchmark problem setup. The cookies are sorted into the containers with the goal of minimizing the number of bins used, the average initial heat of a bin, and the maximum time until the bins can be moved into the pantry.

and the cooling rack has a capacity of R. Also, not every cookie cools down at the same rate due to the presence of flavor particles, such as nuts and chocolate [164], which adds to the problem complexity.

B.4.2.1 Mathematical Formulation

Suppose the baker wants to move the cookies with three goals in mind:

1. minimizing the number of boxes used,
2. minimizing the average initial heat of a box,
3. and minimizing the maximum time until the boxes can be moved to the storefront.

With this time-dependent setup, the dynamic bin packing problem can be formulated. The decision vector shown in (B.14), (B.15), and (B.16) is similar to the

general problem with the addition of the time variable $\mathbf{t_{fill}}$.

$$x_{ij} \in \{0,1\},\ \forall\ i \in \{1,...,\overline{M}\},\ j \in \{1,...,N\} \tag{B.14}$$

$$y_i \in \{0,1\},\ \forall\ i \in \{1,...,\overline{M}\} \tag{B.15}$$

$$t_{fill,i} \in \mathbb{R}^+,\ \forall i \in \{1,...,\overline{M}\} \tag{B.16}$$

The defintions of the i and j indices remain the same, and $t_{fill,i}$ represents the time in seconds at which box i is filled with cookies. However, unlike $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{t}_{fill}$ is a continuous variable, making the dynamic bin packing problem a mixed-variable problem as well.

The mathematical formulation of the dynamic benchmark is given from (B.17) to (B.22).

$$min.\ z_1(s) = \sum_{i=1}^{\overline{M}} y_i \tag{B.17}$$

$$min.\ z_2(s) = \frac{1}{\sum_{i=1}^{\overline{M}} y_i}\left(\sum_{i=1}^{\overline{M}} \frac{1}{(1+r)^{t_{fill,i}}} \sum_{j=1}^{N} hA_s(T_j(t_{fill,i}) - T_\infty)x_{ij}\right) \tag{B.18}$$

$$min.\ z_3(s) = max(t_{available,i},\ i \in \{1,...,\overline{M}\}) \tag{B.19}$$

s.t.

$$\sum_{i=1}^{\overline{M}} x_{ij} = 1,\ \forall j \in \{1,...,N\} \tag{B.20}$$

$$\sum_{j=1}^{N} x_{ij} \leq C y_i,\ \forall i \in \{1,...,\overline{M}\} \tag{B.21}$$

$$\sum_{j=1}^{N} x_{ij} \cdot rack_{ij}(t) \leq R,\ \forall\ t \in \mathbb{R}^+ \tag{B.22}$$

$$\sum_{i \in L_p} y_i \leq F_p,\ \forall\ p \in \{1,...,n_p\} \tag{B.23}$$

$$x_{ij} b_j \Delta t_{batch} < t_{fill,i},\ \forall i \in \{1, ..., \overline{M}\}, \forall j \in \{1, ..., N\} \tag{B.24}$$

The first objective (B.17) remains unchanged, but objectives (B.18) and (B.19) reflect the time-dependence of the cooling cookies. It is assumed that the cookies are placed in a box in a pattern that makes heat transfer interaction between the cookies negligible. Then, the heat in a box is a linear sum of the convective heat produced by each cooling cookie: $hA_s(T_j(t_{fill,i}) - T_\infty)$. In (B.18), T_j is the temperature of cookie j at time $t_{fill,i}$, r is a discount rate, and T_∞ represents the ambient temperature of the room. The discount rate is included in (B.18) to put higher weight on boxes filled earlier to prioritize moving boxes to the storefront during an assumed high-traffic period. It is set at 2.0E-6 for the dynamic test problem. In (B.19), $t_{available,i}$ is the time at which box i is ready to be moved to the storefront.

These objectives are bounded by the "no replacement" constraint (B.20), the box capacity constraint (B.21), the cooling rack capacity constraint (B.22), the period fill limit constraint (B.23), and the "finished baking before boxing" constraint (B.24). The variable F_p is the period fill limit, b_j represents the batch number that cookie j belongs to, and Δt_{batch} is the time required to bake one batch of cookies. In (B.22), $rack_{ij}$ is a binary variable representing if cookie j is present on the cooling rack at a given time, defined in (B.25).

$$rack_{ij}(t) = \begin{cases} 1 & : b_j \Delta t_{batch} \leq t < t_{fill,i} \\ 0 & : \text{otherwise} \end{cases} \tag{B.25}$$

In constraint (B.23), the set L_p is defined by (B.26).

$$L_p = \left\{ i : t_p \leq t_{fill,i} < t_p + \frac{\Delta t_{batch}}{2} \right\} \tag{B.26}$$

Here, t_p marks the beginning of a given time period, and the first set L_0 is defined such that t_0 is equal to Δt_{batch}. The total number of sets n_p depends on the number of half-batch intervals are present until the last box is filled.

Because cookies are small, thin objects, their temperature variation can be modeled using lumped system analysis [165]. Therefore, $T_j(t)$ in (B.18) can be found using (B.27) [166].

$$T(t) = (T_0 - T_\infty)e^{-t/\tau} + T_\infty \tag{B.27}$$

where the time constant τ is defined by (B.28).

$$\tau = \frac{\rho V c_p}{h A_s} \tag{B.28}$$

The temperature of the cookie begins at an initial temperature T_0 upon removal from the oven and cools down to the ambient temperature of the room. The time constant is a combination of the cookie's density ρ, volume V, heat capacity c_p, and surface area A_s and the ambient air's heat transfer coefficient h. It is assumed that the large-scale cookie production has achieved a uniform cookie shape from one batch to the next, so the volume and surface area of each cookie remains constant. It is also assumed that the bakery is operated at room temperature and that the presence of the cookies does not cause variability in the ambient air heat coefficient. Table B.9 lists the values used for these variables. The cookie volume and surface area values were found assuming a diameter of 50 mm and a height of 6 mm.

To find $\mathbf{t}_{available}$ for (B.19), it is assumed that the boxes are ready to be moved once the total heat is equivalent to each cookie in a full box being within 5°C of room temperature. The heat level in a box ready to be moved, Q_{ready}, is given in (B.29).

$$Q_{ready} = hA_s((T_\infty + 5) - T_\infty)C = 5ChA_s \tag{B.29}$$

Table B.9: Thermophysical properties used in the dynamic benchmark.

Property	Given Value
Ambient Air Heat Transfer Coefficient, h $[W/m^2 \cdot K]$	8.0 [166]
Ambient Air Temperature, T_∞ $[^\circ C]/[K]$	25.0 / 298
Cookie Volume, V $[m^3]$	1.2E-5
Cookie Surface Area, A_s $[m^2]$	4.9E-3
Cookie Density, ρ $[kg/m^3]$	1252.3 ± 17.6 [167]
Cookie Specific Heat, c_p $[kJ/kg \cdot K]$	2.94 ± 0.17 [167]
Baking Time, $\Delta t_{batch}[s]$	600

The vector $\mathbf{t}_{available}$ is found using (B.30) to search for the time at which each box i is ready.

$$\sum_{j=1}^{N} hA_s(T_j(t_{available,i}) - T_\infty)x_{ij} = Q_{ready},\ \forall\ i : y_i = 1 \tag{B.30}$$

The modified regula-falsi method is used to find $t_{available,i}$ [141].

To calculate the objectives and constraints in the dynamic problem, every cookie needs to have three characteristics:

- a density ρ,
- a specific heat capacity c_p,
- and a batch number b_j.

As with the static test problem, 2DCPackGen [160] was used to generate the individual cookie characteristics using the settings given in Table B.10. It was assumed that the density and specific heat capacity of an individual cookie were independent char-

Table B.10: Parameter selections to generate experimental samples for the dynamic benchmark problem using 2DCPackGen. When two options are given, the first was set for the small test problem, and the second was set for the large test problem. Only the first half of the output from 2DCPackGen for the small problem was used.

2DCPackGen Option	Selection
Seed	75879 & 758
Number of instances	1
Minimum and maximum size of the large object base	(1000, 2000)
Minimum and maximum size of the small items base	(1, 100)
Minimum and maximum size of the large object height	(1000, 2000)
Minimum and maximum size of the small items height	(1, 1000)
Characteristic of the size and shape of the large object	Big and square
Characteristic of the size and shape of small items	Average size and square
Minimum and maximum number of different item types	(48, 48) & (1000, 1000)
Characteristic of the generator for the height of the large object	Beta distribution w/ $\alpha = 0.5$ and $\beta = 0.5$
Characteristic of the generator for the height of the small items	Beta distribution w/ $\alpha = 1$ and $\beta = 1$
Characteristic of the generator for the number of different item types	Beta distribution w/ $\alpha = 0.5$ and $\beta = 0.5$

acteristics, so 2DCPackGen was initiated using the three-dimensional setting. For the cookie baking process, each box already has a set capacity, so the selections for the large object are irrelevant and are reported here to ensure reproducibility. The values for the small items listed in Table B.10 were carefully chosen for translation into the needed characteristics.

Under the 3-dimensional setting of 2DCPackGen, the characteristic of the width and depth of the small objects were assigned together, while the height was chosen separately. This works well for generating the dynamic data because the density and heat capacity are translated one way while the batch number is translated another way. The batch number for each cookie was found using the generated height characteristic, which was uniformly sampled between 1 and 1000. The generated data was reordered based on this height value, from smallest to largest, and then the batch numbers were assigned in batch-size increments.

To translate the width and depth into densities and heat capacities, it was assumed that these characteristics vary according to a normal distribution. The characteristic of their size and shape was chosen to be average size and square, generated between 1 and 100. Then, the value generated for each of these categories was divided by 100 to become a p-value, which was then translated into a z-value z_j using the normal distribution. From there, the desired characteristic was found using (B.31).

$$q_j = q_{nom} + z_j * \sigma_q \tag{B.31}$$

In (B.31), q_j represents the value of characteristic q for cookie j, nom signifies the nominal value given in Table B.9, and σ_q represents the standard deviation. Table B.9 includes the experimental values for the density and specific heat capacity of cookie dough that was used in the dynamic problem.

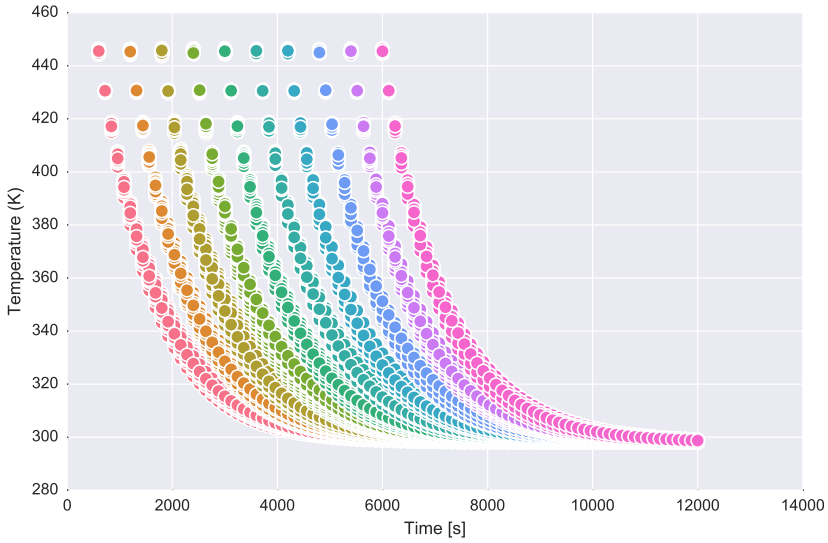


Figure B.4: Example set of 1000 cookies generated for the dynamic benchmark problem. The cookies come out of the oven in batches and cool by convective heat transfer with the ambient air.

Figure B.4 shows a set of 1000 cookies generated using this method. The cookies are created in batches over time and cool down exponentially, as described by (B.27). Due to the variation in density and heat capacity, some cookies cool faster than others, and some cool slower, which is shown in the figure by the broadening of the temperatures during the cooling process.

B.4.2.2 Statistical Setup

To fully demonstrate the new test case, the dynamic bin packing problem was evaluated for both a "toy" problem and a full-size problem. The settings for the levels are given in Table B.11. The full-size problem is large enough to be representative of real-world bin packing problems, and the toy problem is small enough that a brute force investigation of the Pareto front is possible. The results from the toy problem enable a more complete evaluation of the algorithms, and the results from the full

Table B.11: Dynamic Bin Packing Problem settings.

Setting	Toy Problem	Full Problem
Total Number of Cookies N	24	1000
Cookies per Batch	6	100
Box Capacity C [No. of cookies]	8	24
Cooling Rack Capacity R [No. of cookies]	15	300
Period Fill Limit F_p [No. of boxes]	2	8
Parent Population Size	50	100
Function Evaluations	750	25,000

problem illustrate how well the performance scales with problem size.

For the dynamic problem, GAMMA-PC was evaluated against NSGA-II and MOMA. The number of algorithms for comparison was reduced from the static problem based on their performance and the time necessary to translate each algorithm for the dynamic problem. One problem instance was generated for the toy problem, and one for the full problem. The algorithms were run 5 times with different seed values for the toy problem and 20 times with different seeds for the full problem. The basis for the comparison was the number of function evaluations, given in Table B.11.

B.4.3 Performance Metrics

For multiobjective optimization, the task of characterizing the performance of one algorithm over another is not straightforward. It is possible that one approximation set might have higher quality solutions in one area while another is better in a different region of the objective space. One approximation set might have reached

the Pareto set in a very concentrated area of the objective space, while another has slightly lower quality solutions that span the entire Front. Two characteristics highlight these complex differences: the quality of the solutions found and their diversity.

There are many different performance indicators used to describe quality and diversity, but the drawback of many of them is that they require a priori knowledge of the Pareto front. With the test problems used here, the true Pareto front is unknown. This makes it necessary to use indicators that can reliably discern if one set is better than another without knowledge of the Pareto front. This is denoted $A \triangleright B$ and is formally defined in Def. B.2 [129].

Definition B.2. *$A \triangleright B$ if every $z^2 \in B$ is weakly dominated by at least one $z^1 \in A$.*

To prove that the indicator is reliable, it must be shown to be both $\triangleright$-complete and $\triangleright$-compatible [129]. If an indicator is only $\triangleright$-complete, the indicator will always produce a positive result if $A \triangleright B$ but might produce a false positive when $A \not\triangleright B$. If an indicator is only $\triangleright$-compatible, a positive result will always validate that $A \triangleright B$. However, it could produce a false negative, missing cases where $A \triangleright B$. Therefore, a reliable indicator will be both complete and compatible to avoid producing either false positives or false negatives.

Three performance indicators were chosen to evaluate these characteristics. It was previously proven that unary quality indicators cannot be both $\triangleright$-complete and $\triangleright$-compatible [129], so two binary indicators were chosen to quantify the quality of the approximation sets: the binary coverage indicator and the binary ϵ indicator. The maximum spread indicator was chosen to quantify the diversity of the solutions.

B.4.3.1 Binary Coverage Indicator

The first binary quality indicator is the coverage indicator introduced by Zitzler et. al. [168]. This performance metric represents the fraction of one approximation set that is "covered" by another. Equation (B.32) presents the formula to find the coverage of set A over set B, denoted with the symbol $I_C(A, B)$.

$$I_C(A, B) = \frac{|v \in B : \exists\, u \in A \; where \; u \succ v \vee u = v|}{|B|} \tag{B.32}$$

In (B.32), u is an objective vector belonging to approximation set A, and v is an objective vector belonging to B.

By itself, $I_C(A, B)$ shows if approximation set A weakly dominates B, but it needs to be used more carefully to show that A is better than B. To be both $\triangleright$-complete and $\triangleright$-compatible, the indicator needs to pass through interpretation function (B.33).

$$F_C(A \triangleright B) := (I_C(A, B) = 1 \cap I_C(B, A) < 1) \tag{B.33}$$

This logic function is used to analyze the results in the experiments.

B.4.3.2 Binary-ϵ Indicator

The second quality metric is the binary-ϵ indicator introduced by Zitzler et. al. [129]. This indicator is based on the idea quantifying how much one objective vector dominates another, or ϵ-domination, which is defined in Def. B.3.

Definition B.3. *In a multiobjective problem with m objectives, an objective vector $u = (u_1, ..., u_m)^T$ **ϵ-dominates** another vector $v = (v_1, ..., v_m)^T$ iff $\forall \theta \in \{1, ..., m\}$, $u_\theta \leq \epsilon \cdot v_\theta$ for a given $\epsilon > 0$, written $u \succeq_\epsilon v$.*

To apply this idea to approximation sets, a series of calculations are made [129].

First, each solution belonging to approximation set A are compared to each solution in B using (B.34).

$$\epsilon_{z^1,z^2} = \max_{1\leq\theta\leq n} \frac{z^1_\theta}{z^2_\theta},\ \forall\ z^1 \in A, z^2 \in B \tag{B.34}$$

During the comparison, ϵ_{z^1,z^2} is set as the maximum fraction among the objectives. Since this is a minimization problem, the maximum fraction indicates the least dominated objective and therefore the ϵ-domination of z^1 to z^2. Then, all of the ϵ_{z^1,z^2} values for a particular solution belonging to set B are compared using (B.35).

$$\epsilon_{z^2} = \min_{z^1\in A} \epsilon_{z^1,z^2},\ \forall\ z^2 \in B \tag{B.35}$$

The variable ϵ_{z^2} indicates how well a particular solution in set B is dominated by set A overall. Finally, the binary-ϵ indicator $I_\epsilon(A, B)$ is set as the worst of these values, as shown in (B.36).

$$I_\epsilon(A, B) = \max_{z^2\in B} \epsilon_{z^2} \tag{B.36}$$

Combining these steps produces (B.37).

$$I_\epsilon(A, B) = \max_{z^2\in B} \min_{z^1\in A} \max_{1\leq i\leq n} \frac{z^1_\theta}{z^2_\theta} \tag{B.37}$$

As with the binary coverage indicator, $I_\epsilon(A, B)$ must pass through an interpretation function to be both $\triangleright$-complete and $\triangleright$-compatible. Function (B.38) shows the logic used to prove that approximation set A is better than B.

$$F_\epsilon(A \triangleright B) := (I_\epsilon(A, B) \leq 1 \cap I_\epsilon(B, A) > 1) \tag{B.38}$$

B.4.3.3 Maximum Spread Indicator

The maximum spread indicator is used to compare the diversity of the approximation sets [169]. This performance metric measures how far an approximation set reaches in the objective space along the different objective vertices. It doesn't provide evidence that one set is better than another, but it does describe how well an algorithm approximates the front. The spread D for one approximation set is calculated using (B.39).

$$D = \left[\sum_{\theta=1}^{m} \left(\max_{z \in A} \{z_\theta\} - \min_{z \in A} \{z_\theta\} \right)^2 \right]^{1/2} \tag{B.39}$$

Adra and Flemin suggest normalizing D values by the spread of the true Pareto front [169], but that is not possible here. Instead, the spread indicator values are normalized using the largest D-value found every experiment.

B.4.3.4 Pareto Front Performance Metrics

When the Pareto front is known, more straightforward metrics may be used. The absolute efficiency of each algorithm in finding solutions that belong to the Pareto set is defined in (B.40) [147].

$$E = \frac{|A \cap OP|}{|OP|} \tag{B.40}$$

Here, E is the proportion of approximate set A that belongs to the Pareto set OP, showing how well each algorithm has reached optimal solutions. However, this measure could return the same efficiency to two different approximate sets, even if one set achieved a front that was much closer to the Pareto front than the other. Therefore, it is useful to complement the efficiency with measures describing the distance from an approximate set to the Pareto set.

The distance $d(u, v)$ between two solutions is defined by the Tchebycheff norm,

given in (B.41).

$$d(u, v) = \sum_{\theta=1}^{m} \lambda_\theta |f_\theta(u) - f_\theta(v)| \tag{B.41}$$

Here, the distance is calculated in the objective space, summing the difference between the fitness values for each objective θ. The variable λ_θ is a parameter to normalize each objective to the others. The distance between an individual solution and the Pareto set is then the minimum distance to any solution belonging to the Pareto set [147].

Three distance indicators can be combined with the absolute efficiency to evaluate the algorithms with respect to the Pareto front [147]:

- the maximum distance between sets A and OP,
- the average distance from set A to set OP,
- and the pooled standard deviation σ_{pool} of the distance, defined in (B.42).

$$\sigma_{pool} = \sqrt{\frac{\sum_{f=1}^{S}(n_f - 1)s_f}{\sum_{f=1}^{S} n_f}} \tag{B.42}$$

Here, n_f is the size of sample f, s_f is the unbiased sample standard deviation, and S is the total number of samples collected. The pooled standard deviation is preferred over considering the individual sample standard deviations due to a higher statistical power.

B.4.3.5 Empirical Attainment Function

The empirical attainment function was used to show the objective space of the dynamic problems. It is a generalization of the multivariate cumulative distribution function and combines a number of approximate sets to form one function [170]. It is defined as the probability that the combined sets will attain an arbitrary point in the objective space, and the algorithm to calculate the 3-dimensional version of the

function is given in [171]. The probability is denoted using "%-attainment surfaces," where each point along the surface has the given probability of existing in one of the approximation sets produced by an algorithm.

The underlying structure of the 3-dimensional algorithm is a balanced binary search tree to ensure that the empirical attainment function is calculated in $O(n^2 m \log(m))$ time. The implementation of the balanced binary search tree to support the 3-dimensional empirical attainment algorithm in this study was based on [172]. The resulting graph can be used to explore how well each algorithm performs in specific areas of the objective space.

B.5 Simulation Results

B.5.1 Static Problem Results

GAMMA-PC performed at least as well as the other state-of-the-art MOEAs in the static bin packing problems. The static performance metrics are illustrated in Figs. B.5, B.6, and B.7. Fig. B.5 presents box-and-whisker plots of the binary coverage indicator $I_C(A, B)$, comparing the five algorithms in pairs. A higher coverage value indicates a better outcome, so the box and whiskers for GAMMA-PC suggest that it tends to cover more of the other approximation sets.

Table B.12 presents the statistical analysis of Fig. B.5, using an overall Type-I error rate of 0.05% and Bonferroni's Method [173] to evaluate if each comparison meets the necessary condition of $(I_C(A, B) = 1 \cap I_C(B, A) < 1)$. This is determined by two tests. The Wilcoxon-rank sum test is performed first to determine if the samples in the comparison belong to the same distribution. If they are determined to be different, the student-t difference test is then performed to determine if the difference between the two is effectively the difference between 1 and the lower average I_C value, assuming similar standard deviations. Table B.12 shows that while the I_C

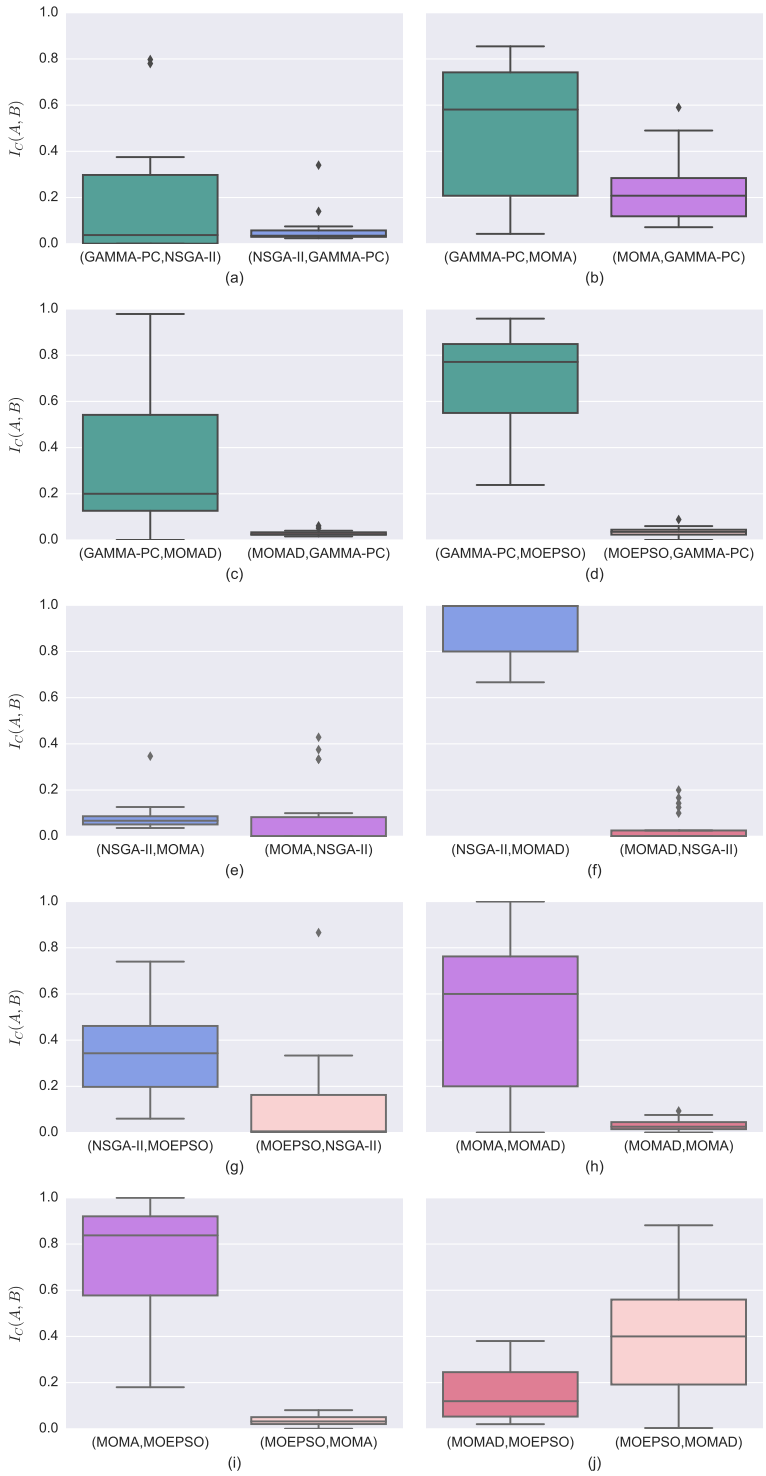


Figure B.5: Box plots of the binary coverage indicators comparing GAMMA-PC, NSGA-II, MOMA, MOMAD, and MOEPSO, based on 20 experimental runs.

Table B.12: Statistical evaluation if GAMMA-PC performs better than NSGA-II, MOMA, MOMAD, or MOEPSO according to the $I_C(A, B)$ metric. First, the Wilcoxon-rank sum test is used to determine if the samples belong to the same distribution, with the null hypothesis $H_{wr,0} : I_C(A, B) = I_C(B, A)$ and the alternative hypothesis $H_{wr,1} : I_C(A, B) \neq I_C(B, A)$. Then, if $H_{wr,1}$ is accepted, the student-t test is applied to determine if the difference between the samples corresponds to the null hypothesis $H_{st,0} : I_C(A, B) - I_C(B, A) \geq 1.0 - I_C(B, A)$ or to the alternative hypothesis $H_{st,1} : I_C(A, B) - I_C(B, A) < 1.0 - I_C(B, A)$. The degrees of freedom was 38, and the significance level for each test was set at 0.0125 to ensure an overall Type I error rate of 5% for the comparisons.

	Wilcoxon-Rank Sum Test	Student-t Difference Test
A = GAMMA-PC, B = NSGA-II		
test statistic	-0.35	–
p-value	0.73	–
Accepted Hypothesis	$H_{wr,0}$	–
A = GAMMA-PC, B = MOMA		
test statistic	2.5	–
p-value	0.014	–
Accepted Hypothesis	$H_{wr,0}$	–
A = GAMMA-PC, B = MOMAD		
test statistic	3.2	-8.9
p-value	0.0012	<0.0001
Accepted Hypothesis	$H_{wr,1}$	$H_{st,1}$
A = GAMMA-PC, B = MOEPSO		
test statistic	5.4	-6.1
p-value	<0.0001	<0.0001
Accepted Hypothesis	$H_{wr,1}$	$H_{st,1}$

values for GAMMA-PC were proven to be significantly different than the values for MOMAD and MOEPSO, GAMMA-PC's performance cannot be proven to be significantly better (or worse) than NSGA-II, MOMA, MOMAD, or MOEPSO by the binary coverage indicator.

Fig. B.6 presents box-and-whisker plots of the binary-ϵ indicator $I_\epsilon(A, B)$. A lower value indicates a better outcome, so the boxes for GAMMA-PC suggests that it performs well by this metric. The comparison to MOMA in Fig. B.6(b) shows approximately the same range of values for I_ϵ, but GAMMA-PC's range tightly hugs a value of 1.0 for the other three comparisons. Table B.13 presents the statistical analysis of Fig. B.6, using an overall Type I error rate of 0.05% and Bonferroni's Method to evaluate if each binary comparison meets the condition of $(I_\epsilon(A, B) \leq 1 \cap I_\epsilon(B, A) > 1)$. With this metric, GAMMA-PC is proven to perform better than NSGA-II, MOMAD, and MOEPSO but not MOMA.

The comparisons between the other algorithms are also illustrated in Figs. B.5 and B.6, although they were not evaluated statistically. Fig. B.5(f) suggests that NSGA-II likely performs better than MOMAD, and Fig. B.5(i) suggests that MOMA likely performs better than MOEPSO. Figs. B.6(e), (h), and (i) also suggest that MOMA performs better NSGA-II, MOMAD, and MOEPSO. However, the majority of the binary comparisons do not indicate which algorithm performs better for the static problem. This means either the static bin packing problem presented little difficulty, or the algorithms produce similar levels of performance.

Fig. B.7 presents the normalized spread indicators for the five algorithms on a logrithmic scale. A higher value indicates greater diversity, so this figure indicates that GAMMA-PC produces the most diversity in its approximation sets, followed closely by MOMA. The difference between the spread values for these two is small but statistically significant (refer to Appendix). The approximation sets for NSGA-

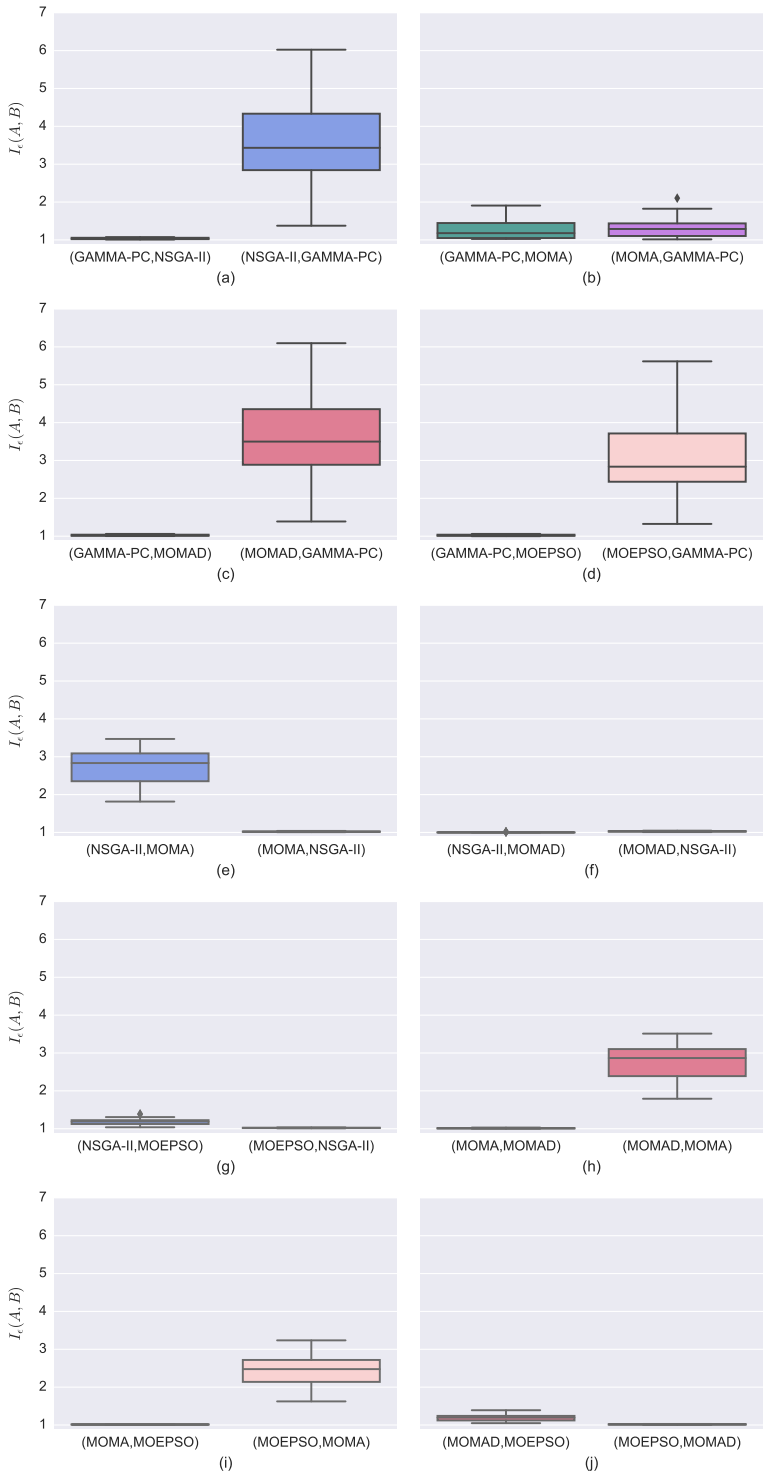


Figure B.6: Box plots of the binary-ϵ indicators comparing GAMMA-PC, NSGA-II, MOMA, MOMAD, and MOEPSO, based on 20 experimental runs.

Table B.13: Statistical evaluation if GAMMA-PC performs better than NSGA-II, MOMA, MOMAD, or MOEPSO according to the $I_\epsilon(A, B)$ metric. First, the Wilcoxon-rank sum test is used to determine if the samples belong to the same distribution, with the null hypothesis $H_{wr,0} : I_\epsilon(A, B) = I_\epsilon(B, A)$ and the alternative hypothesis $H_{wr,1} : I_\epsilon(A, B) \neq I_\epsilon(B, A)$. Then, if $H_{wr,1}$ is accepted, the student-t test is applied to determine if the difference between the samples corresponds to the null hypothesis $H_{st,0} : I_\epsilon(B, A) - I_\epsilon(A, B) \geq I_\epsilon(B, A) - 1.0$ or to the alternative hypothesis $H_{st,1} : I_\epsilon(B, A) - I_\epsilon(A, B) < I_\epsilon(B, A) - 1.0$. The degrees of freedom was 38, and the significance level for each test was set at 0.0125 to ensure an overall Type I error rate of 5% for the comparisons.

	Wilcoxon-Rank Sum Test	Student-t Difference Test
A = GAMMA-PC, B = NSGA-II		
test statistic	-5.4	-0.11
p-value	<0.0001	0.46
Accepted Hypothesis	$H_{wr,1}$	$H_{st,0}$
A = GAMMA-PC, B = MOMA		
test statistic	-0.54	–
p-value	0.59	–
Accepted Hypothesis	$H_{wr,0}$	–
A = GAMMA-PC, B = MOMAD		
test statistic	-5.4	-0.08
p-value	<0.0001	0.46
Accepted Hypothesis	$H_{wr,1}$	$H_{st,0}$
A = GAMMA-PC, B = MOEPSO		
test statistic	-5.4	-0.09
p-value	<0.0001	0.47
Accepted Hypothesis	$H_{wr,1}$	$H_{st,0}$

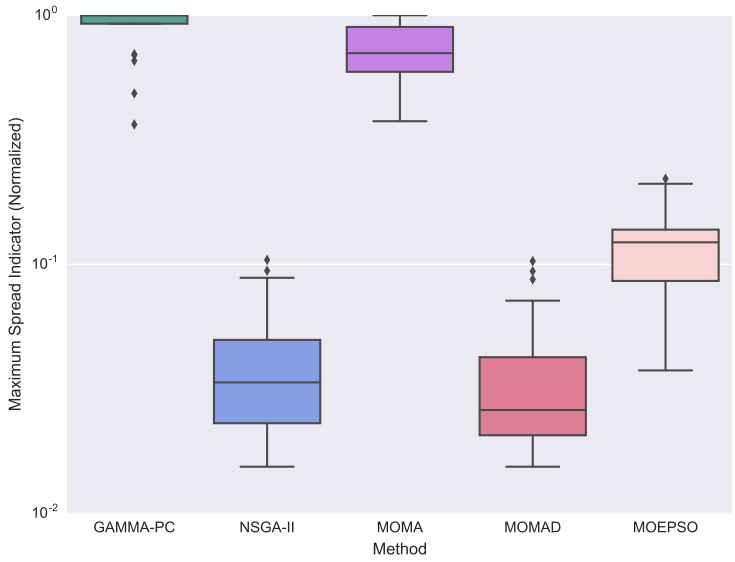


Figure B.7: Normalized spread indicators for static benchmark experiments on a log-scale. The indicators were normalized based on the largest spread in each experiment.

II and MOMAD exhibit similar levels of diversity, while those for MOEPSO have a somewhat higher spread.

While GAMMA-PC produced mainly highly diverse, quality solutions, it did not perform as well in a few of the bin packing scenarios that 2DCPackGen [160] generated. Fig. B.8 illustrates a typical example of the approximation set produced by GAMMA-PC. In the figure, the GAMMA-PC set extends across the objective space, finding either similar or better solutions than the other algorithms. In constrast, Fig. B.9 presents one of the aberrant results. In this second graph, GAMMA-PC did not produce an approximation set as diverse as the set produced by MOMA and produced inferior solutions above about 120 bins on the x-axis.

The illustrations of the objective space in Figs. (B.8) and (B.9) also give an indication of why GAMMA-PC was proven to perform better than NSGA-II, MOMAD,

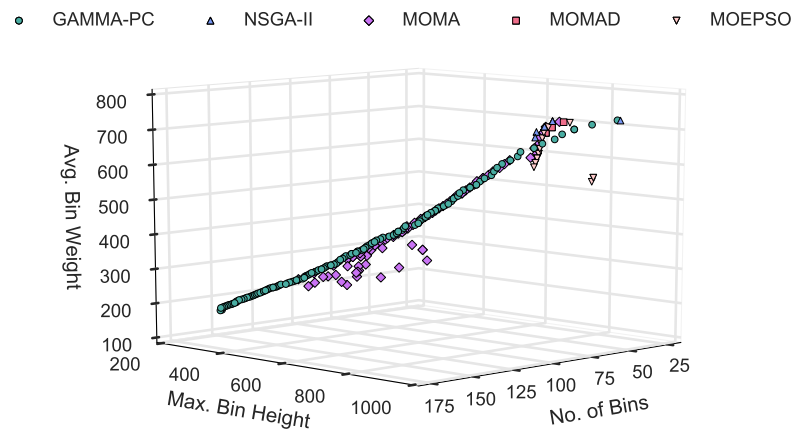


Figure B.8: Three-dimensional scatterplot of the approximate sets produced for experiment 1 of the static problem.

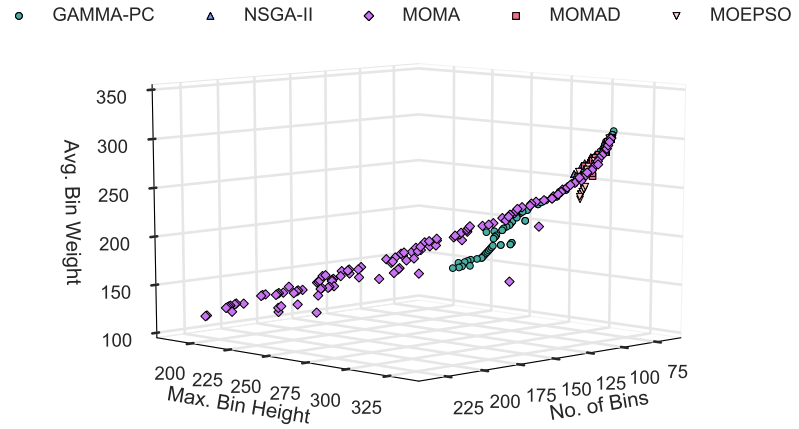


Figure B.9: Three-dimensional scatterplot of the approximate sets produced for experiment 8 of the static problem.

and MOEPSO using the binary I_ϵ indicator but not with the I_C indicator. While the approximation sets for GAMMA-PC span the objective space, the solutions with small numbers of bins in use were not as well exploited as they were by these three algorithms. The solutions for NSGA-II, MOMAD, and MOEPSO were highly concentrated at this lower end in all of the experiments, suggesting that their algorithms were stuck in this region. While this negatively impacted their diversity, the algorithms were able to find better solutions in this region. The coverage of these sets is therefore lower. The I_ϵ indicator is found based proportions of individual fitness values across the entire objective space, so GAMMA-PC performs better by this metric.

B.5.2 Toy Dynamic Problem Results

Based on the poor performance of MOMAD and MOEPSO for the static problem, only NSGA-II and MOMA were used to evaluate GAMMA-PC in the dynamic bin packing problems. For the toy problem, the Pareto front was found by brute force calculations, evaluating as many box combinations as possible. The Pareto front shown in Fig. B.10 is the best-known front found for the toy dynamic bin packing problem. It covers a range of solutions, from those with boxes filled to capacity to those with nearly empty boxes. As the number of boxes increases, the average initial heat decreases approximately exponentially, and the maximum time to move decreases approximately linearly. The two latter objectives are not competing values, although they are not purely directly correlated. As the initial box heat increases, the maximum time to move curves upward, reducing its slope with higher heat levels. This front was used to evaluate the algorithms by their absolute efficiency and the distance to the front in the objective space.

Fig. B.11 presents the absolute efficiency of GAMMA-PC, NSGA-II, and MOMA

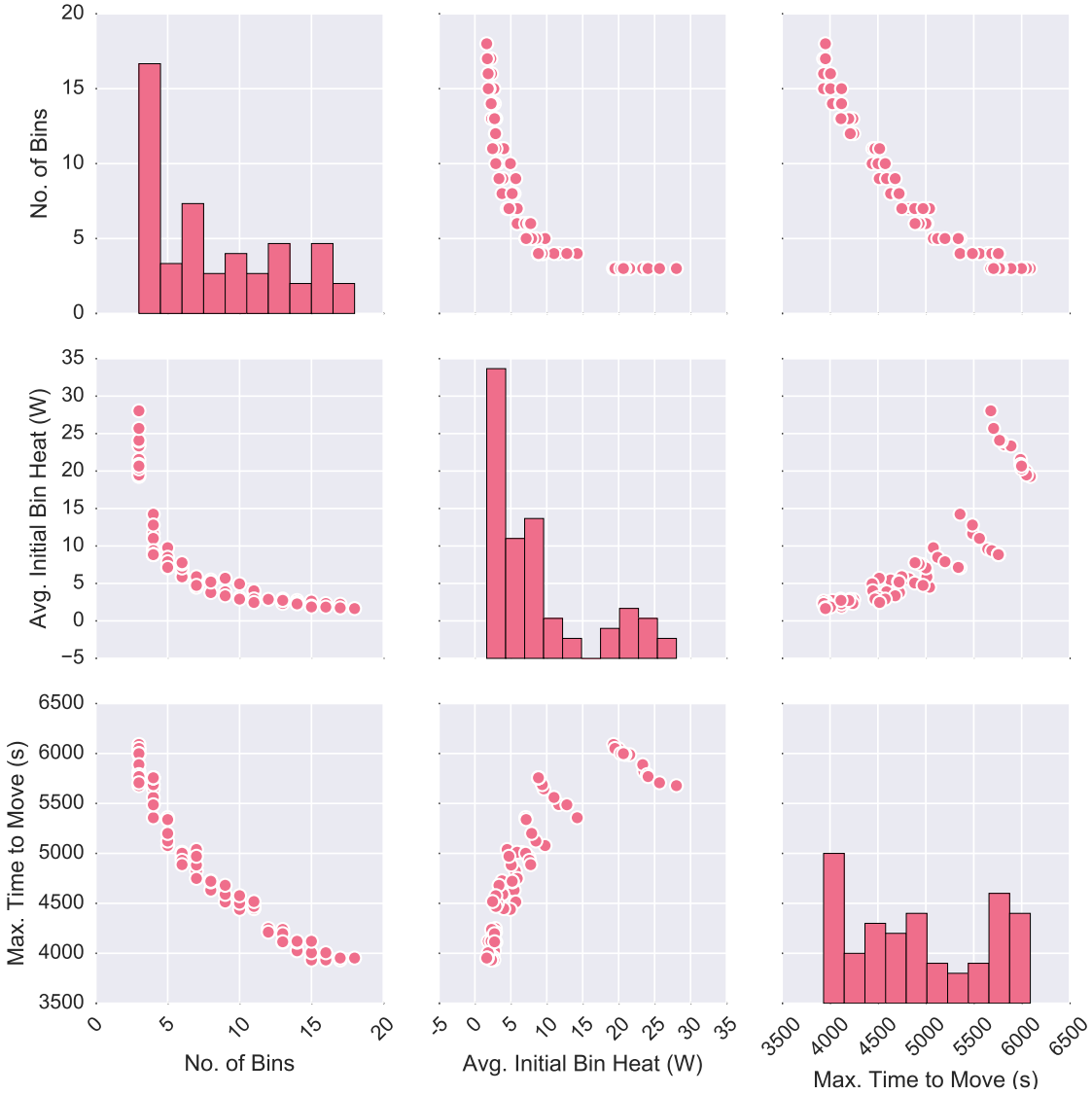


Figure B.10: Best-known Pareto front found for the toy dynamic problem.

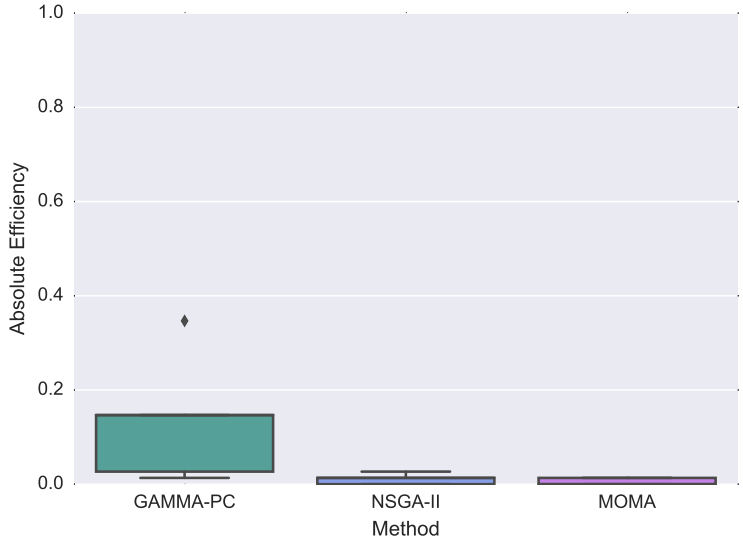


Figure B.11: Box plot of the absolute efficiency of each algorithm compared to the best-known Pareto front.

for the toy dynamic problem. All three algorithms present low levels of efficiency in terms of the Pareto front. Although GAMMA-PC shows the largest range with an outlier at 34%, its absolute efficiency is not significantly different from the other two (refer to Appendix). This is due to the small sample size and the presence of two poorly performing runs that overlapped the ranges for NSGA-II and MOMA.

Fig. B.12 shows box-and-whisker plots of the average distance to the Pareto front on the left and the maximum distance on the right. The approximation sets produced by GAMMA-PC are shown to be much closer to the Pareto front than those for NSGA-II or MOMA. While the range of the maximum distance for GAMMA-PC overlaps the lower part of the ranges for NSGA-II and MOMA, the decrease in distance is statistically significant (refer to Appendix). The average distance from approximation sets produced by GAMMA-PC to the Pareto front is also much smaller than the distance for NSGA-II or MOMA.

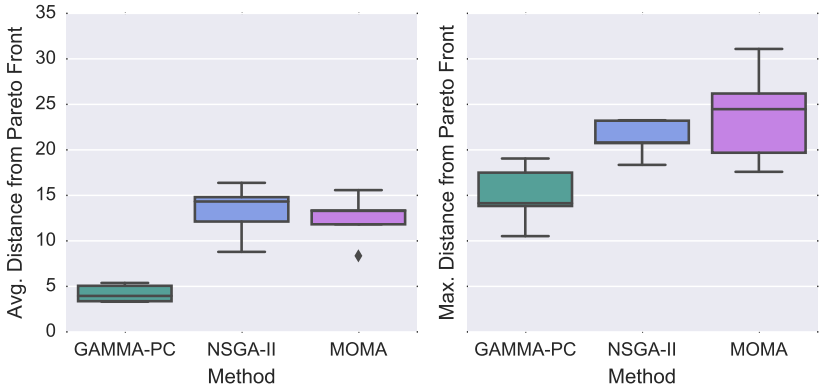


Figure B.12: Box plot of the average and maximum distance to the best-known Pareto front for each algorithm, based on 5 experimental runs.

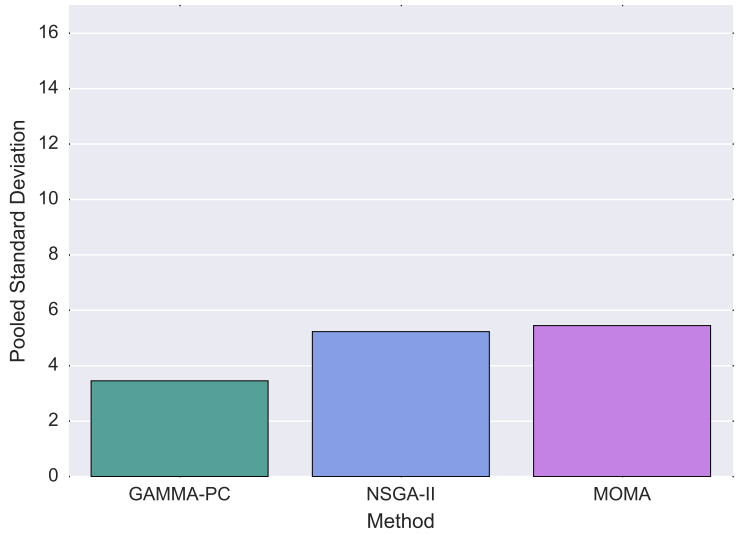


Figure B.13: Bar graph of the pooled standard deviation of the distance to the best-known Pareto front for each algorithm, pooled from results of 5 experimental runs.

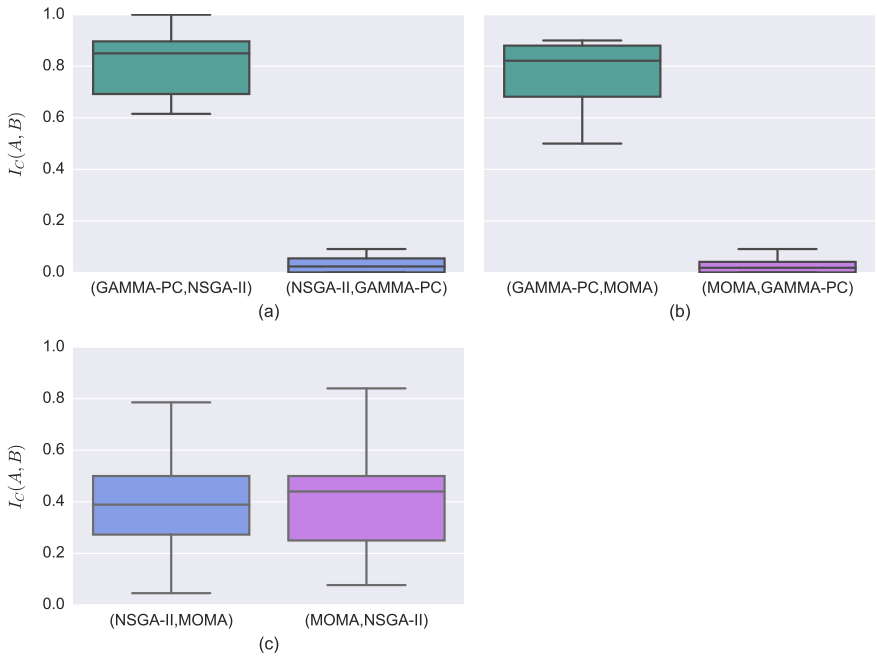


Figure B.14: Box plots of the binary coverage indicators comparing (a) GAMMA-PC to NSGA-II, (b) GAMMA-PC to MOMA, and (c) NSGA-II to MOMA, based on 5 experimental runs.

Fig. B.13 shows the pooled standard deviation of the distance to the Pareto front. NSGA-II and MOMA produce approximation sets with about the same standard deviation, while the approximation sets produced by GAMMA-PC have a lower standard deviation. Since the GAMMA-PC approximation sets are also much closer to the Pareto front, this suggests that the majority of the solutions found by GAMMA-PC lay close to that average distance away.

Fig. B.14 presents box-and-whisker plots of the binary coverage indicator I_C. While only 5 experimental runs were completed for the toy problem, each box represents a sample of 25 I_C values since all of the computations were done for the same problem with different seed values. Therefore, each approximation set produced by one algorithm was compared to all of the approximation sets produced by

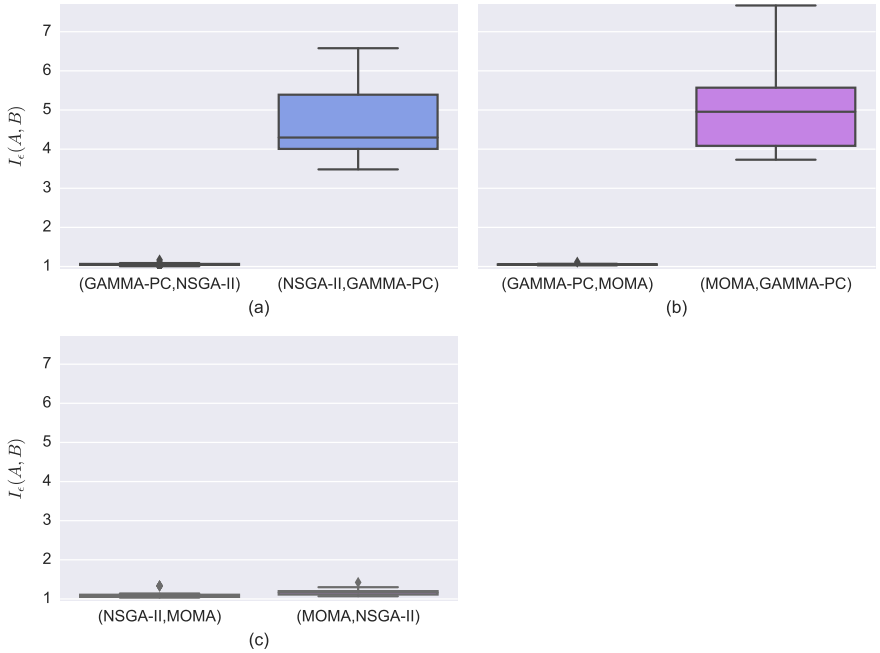


Figure B.15: Box plots of the binary-ϵ indicators comparing (a) GAMMA-PC to NSGA-II, (b) GAMMA-PC to MOMA, and (c) NSGA-II to MOMA, based on 5 experimental runs.

the other algorithm. NSGA-II and MOMA exhibit nearly indentical box plots of I_C values, but GAMMA-PC produced approximation sets with much higher coverage values for the toy dynamic problem. Table B.14 presents the statistical evaluation of the interpretation function for I_C, using the same statistical analysis discussed in Sec. B.5.1. While it is proven that the values for GAMMA-PC belong to separate distributions than its counterparts, it cannot be proven that (I_C(GAMMA-PC, B) = $1 \cap I_C$(B,GAMMA-PC) < 1) for either NSGA-II or MOMA. Therefore, even though GAMMA-PC approximation sets do cover more of the other sets, its performance cannot be proven to be strictly better or worse than NSGA-II or MOMA by the binary coverage indicator.

Fig. B.15 presents box-and-whiskers plots of the binary-ϵ indicator I_ϵ. The values

Table B.14: Statistical evaluation if GAMMA-PC performs better than NSGA-II or MOMA according to the $I_C(A, B)$ metric. First, the Wilcoxon-rank sum test is used to determine if the samples belong to the same distribution, with the null hypothesis $H_{wr,0} : I_C(A, B) = I_C(B, A)$ and the alternative hypothesis $H_{wr,1} : I_C(A, B) \neq I_C(B, A)$. Then, if $H_{wr,1}$ is accepted, the student-t test is applied to determine if the difference between the samples corresponds to the null hypothesis $H_{st,0} : I_C(A, B) - I_C(B, A) \geq 1.0 - I_C(B, A)$ or to the alternative hypothesis $H_{st,1} : I_C(A, B) - I_C(B, A) < 1.0 - I_C(B, A)$. The degrees of freedom was 48 for each test, and the significance level for each test was set at 0.016 to ensure an overall Type I error rate of 5% for the comparisons.

	Wilcoxon-Rank Sum Test	Student-t Difference Test
A = GAMMA-PC, B = NSGA-II		
test statistic	6.1	-7.5
p-value	<1.0E-5	<1.0E-5
Accepted Hypothesis	$H_{wr,1}$	$H_{st,1}$
A = GAMMA-PC, B = MOMA		
test statistic	6.1	-8.8
p-value	<1.0E-5	<1.0E-5
Accepted Hypothesis	$H_{wr,1}$	$H_{st,1}$
A = MOMA, B = NSGA-II		
test statistic	-0.5	–
p-value	0.62	–
Accepted Hypothesis	$H_{wr,0}$	–

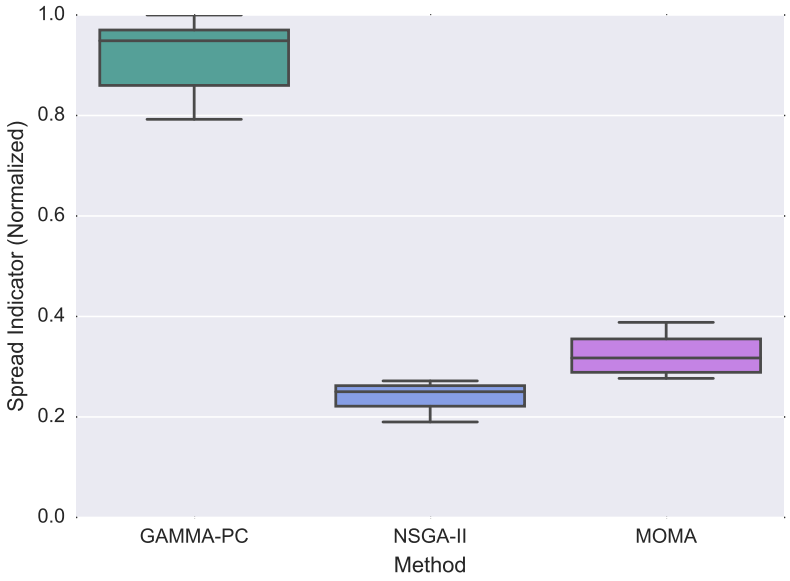


Figure B.16: Box plot of the maximum spread values for each algorithm over 5 experimental runs.

for the GAMMA-PC results show a much lower range than those for NSGA-II or MOMA, closely hugging the 1.0 I_ϵ value that suggests better performance. Again, the I_ϵ values for comparing NSGA-II and MOMA seem to show almost indentical ranges. Table B.15 presents the statistical evaluation of Fig. B.15. The samples of I_ϵ values comparing GAMMA-PC to NSGA-II and MOMA were both proven to belong to different distributions than their counterparts and to meet the necessary condition of $(I_\epsilon(\text{GAMMA-PC}, B) \leq 1 \cap I_\epsilon(B,\text{GAMMA-PC}) > 1)$. Comparing NSGA-II to MOMA showed that the difference in their I_ϵ values was statistically significant, but neither met the necessary condition to show better performance.

The diversity of the solutions found by GAMMA-PC are more pronounced in the toy dynamic problem than they were in the static problem. Fig. B.16 shows box-and-whisker plots of the maximum spread indicator. The difference in the spread values

Table B.15: Statistical evaluation if GAMMA-PC performs better than NSGA-II or MOMA according to the $I_\epsilon(A,B)$ metric. First, the Wilcoxon-rank sum test is used to determine if the samples belong to the same distribution, with the null hypothesis $H_{wr,0} : I_\epsilon(A,B) = I_\epsilon(B,A)$ and the alternative hypothesis $H_{wr,1} : I_\epsilon(A,B) \neq I_\epsilon(B,A)$. Then, if $H_{wr,1}$ is accepted, the student-t test is applied to determine if the difference between the samples corresponds to the null hypothesis $H_{st,0} : I_\epsilon(B,A) - I_\epsilon(A,B) \geq I_\epsilon(B,A) - 1.0$ or to the alternative hypothesis $H_{st,1} : I_\epsilon(B,A) - I_\epsilon(A,B) < I_\epsilon(B,A) - 1.0$. The degrees of freedom was 48 for each test, and the significance level for each test was set at 0.016 to ensure an overall Type I error rate of 5% for both comparisons.

	Wilcoxon-Rank Sum Test	Student-t Difference Test
A = GAMMA-PC, B = NSGA-II		
test statistic	-6.1	-0.28
p-value	<1.0E-5	0.39
Accepted Hypothesis	$H_{wr,1}$	$H_{st,0}$
A = GAMMA-PC, B = MOMA		
test statistic	-6.1	-0.24
p-value	<1.0E-5	0.41
Accepted Hypothesis	$H_{wr,1}$	$H_{st,0}$
A = MOMA, B = NSGA-II		
test statistic	-3.2	-4.5
p-value	1.4E-3	<1.0E-5
Accepted Hypothesis	$H_{wr,1}$	$H_{st,1}$

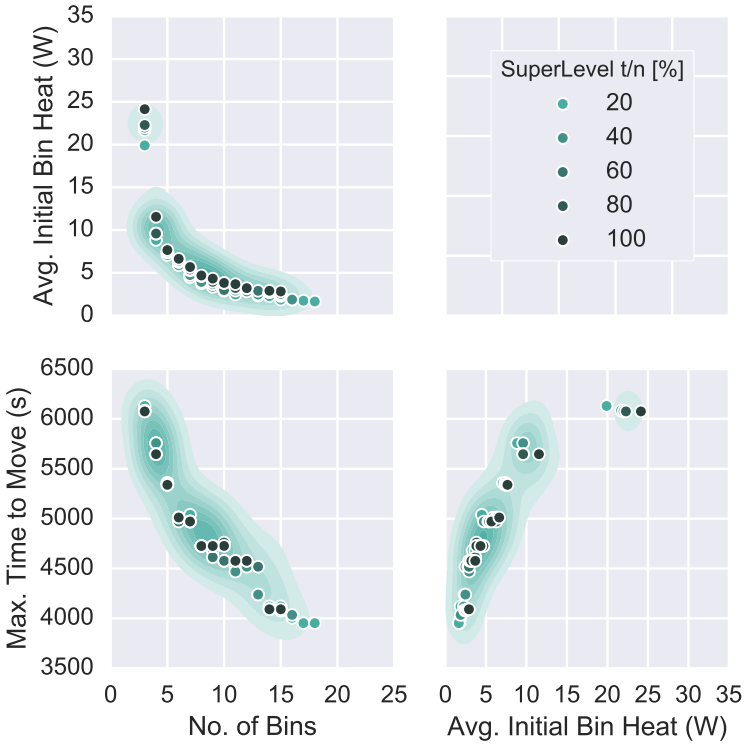


Figure B.17: Two-dimensional scatter matrix of the 3-dimensional empirical attainment function produced by GAMMA-PC for the toy dynamic problem.

shown here is statistically significant (refer to Appendix). The spread indicators for GAMMA-PC are more than two times larger than those of NSGA-II or MOMA. While the difference between GAMMA-PC and NSGA-II is smaller than in the static problem, the difference between GAMMA-PC and MOMA grew. The ranges for the maximum spread indicator for NSGA-II and MOMA are much more similar as well.

Overall, NSGA-II and MOMA behaved more similarly in the toy dynamic problem than in the static problem, while GAMMA-PC performed about as well as before. A consideration of the objective space shows that NSGA-II and MOMA explored the same part of the map. A partial two-dimensional scatter matrix plot of the empirical attainment function for GAMMA-PC, NSGA-II, and MOMA are shown

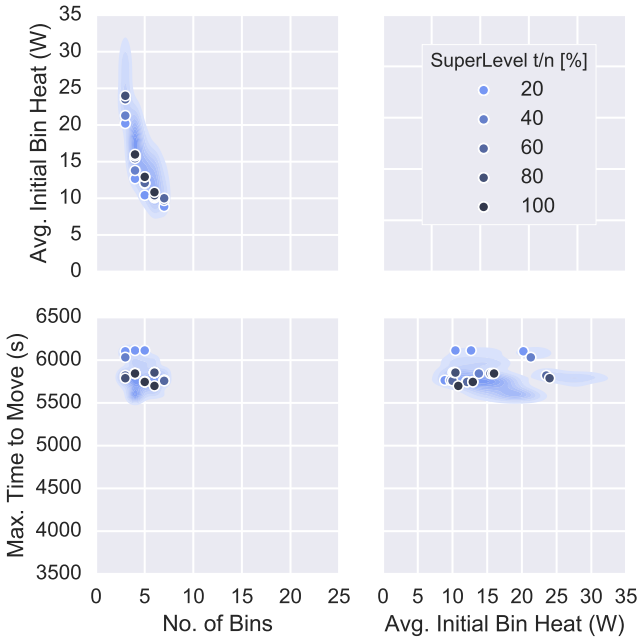


Figure B.18: Two-dimensional scatter matrix of the 3-dimensional empirical attainment function produced by NSGA-II for the toy dynamic problem.

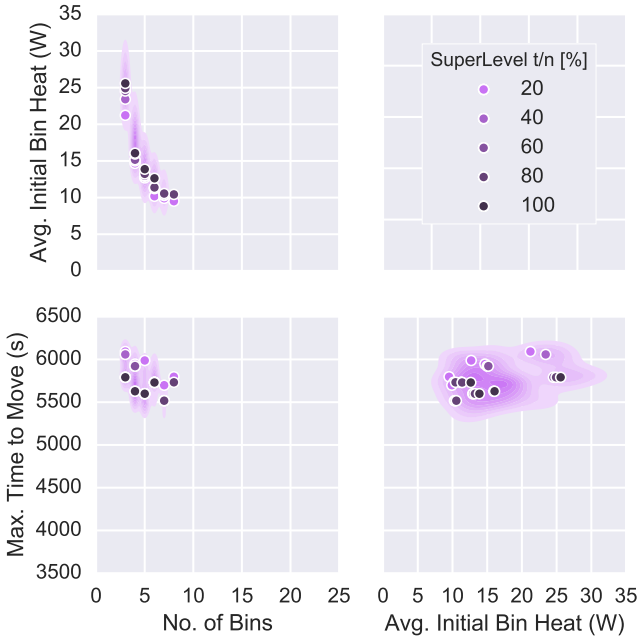


Figure B.19: Two-dimensional scatter matrix of the 3-dimensional empirical attainment function produced by MOMA for the toy dynamic problem.

in Figs. B.17, B.18, and B.19. Each plot shows points along the 20%, 40%, 60%, 80%, and 100% attainment surfaces of the empirical attainment function overlaid on a kernel density estimate of all the nondominated solutions found by each algorithm. The 20% surface indicates solutions found in only 20% of the experimental runs, the 40% surface indicates those found in 40% of the runs, and so on.

These plots show GAMMA-PC produced approximation sets that look much more like the Pareto front in Fig. B.10 than NSGA-II or MOMA did. Fig. B.17 shows that GAMMA-PC produced solutions in a much broader range along the x- and z-axis (No. of Bins and Max. Time to Move, respectively). The range along the y-axis is also shifted toward smaller values than is shown in Figs. B.18 and B.19. The figures also indicate that NSGA-II and MOMA produced solutions with low numbers of bins in use, so the shift in higher initial heat values is reasonable. However, the difference in the maximum time to move to the storefront is not entirely explained by the concentration at the low end of the x-axis, as the attainment surface for GAMMA-PC shows much lower maximum time values above about 5 bins, which was an area explored by both NSGA-II and GAMMA-PC.

As with the static problem, it makes sense that GAMMA-PC was proven to perform better by the binary I_ϵ indicator but not by the binary I_C indicator. Both NSGA-II and MOMA produced solutions at the very low end of the x-axis that were neither dominated or equal to the solutions produced by GAMMA-PC in this area. The 20% attainment surface shown in Fig. B.17 includes a point $x = 3$ bins, $y = 20$ W, which matches the corresponding 20% attainment point in Fig. B.18 and is lower than the 20% point in Fig. B.19. However, this point in the 40% attainment surface for GAMMA-PC has a higher initial heat value than the corresponding 40% attainment point for NSGA-II. While GAMMA-PC may have found solutions with lower maximum time values, the solutions with lower average initial heat values found by

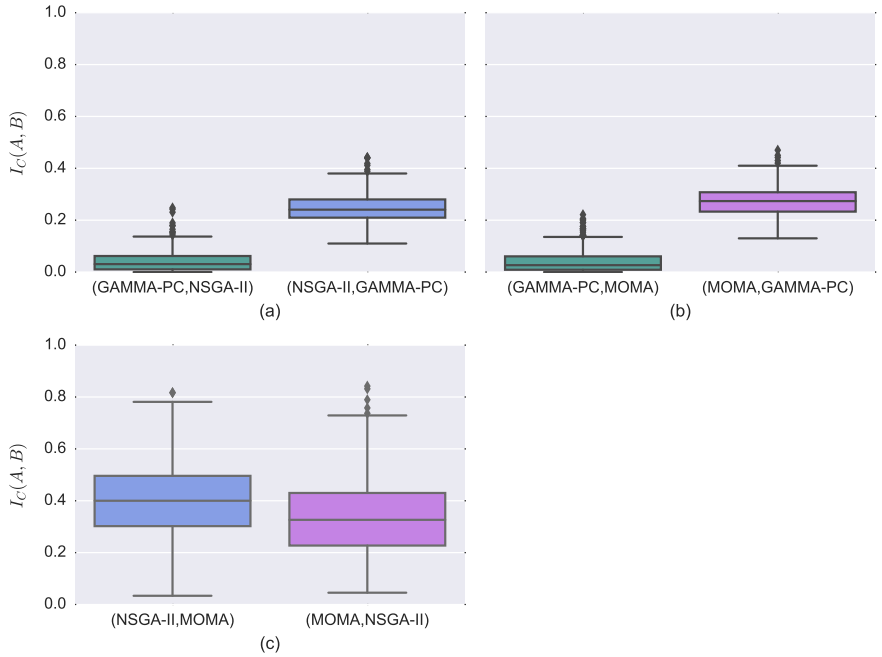


Figure B.20: Box plots of the binary coverage indicators comparing (a) NSGA-II to MOMA, (b) NSGA-II to GAMMA-PC, and (c) MOMA to GAMMA-PC, based on 20 experimental runs.

NSGA-II in this region are nondominated by GAMMA-PC. This relationship is also seen with MOMA to a lesser extent. Therefore, the approximation sets produced by GAMMA-PC do not completely cover the others in enough experiments to perform better by the I_C metric. At the same time, the solutions produced by GAMMA-PC are much closer to the Pareto front along the entire objective space, so it does perform better by the I_ϵ metric.

B.5.3 Full Dynamic Problem Results

GAMMA-PC did not perform as well in the full dynamic problem as it did in the toy dynamic problem. While it maintained a higher diversity of solutions, the quality of those solutions were lower. Fig. B.20 shows the binary coverage indicators I_C for the three comparisons. Since 20 experimental runs were completed for each

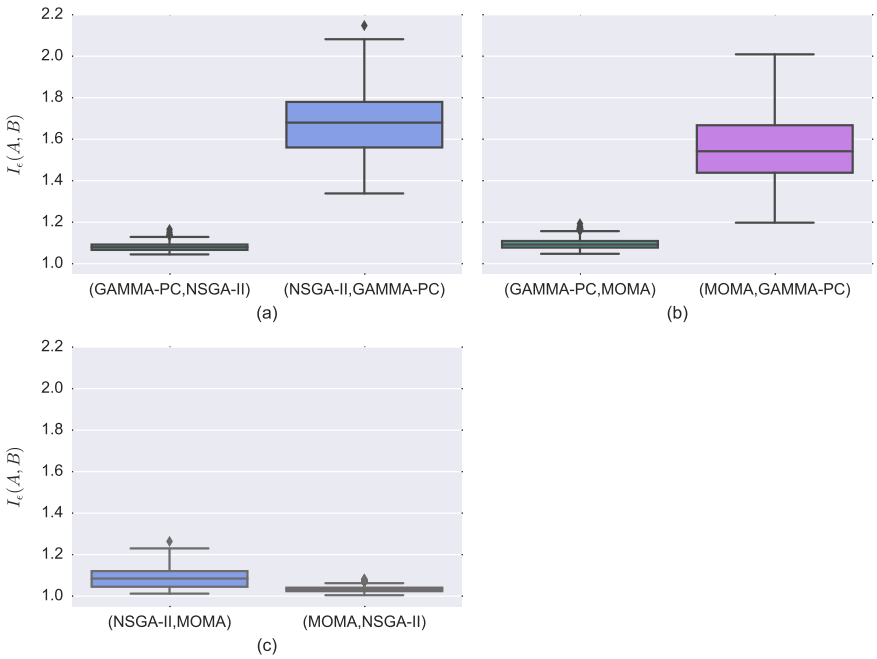


Figure B.21: Box plots of the binary-ϵ indicators comparing (a) NSGA-II to MOMA, (b) NSGA-II to GAMMA-PC, and (c) MOMA to GAMMA-PC, based on 20 experimental runs.

algorithm, there are 400 I_C values in each box-and-whisker. GAMMA-PC is shown to have lower coverage over the sets produced by NSGA-II and MOMA, although Table B.16 proves that neither performs better than GAMMA-PC by this metric. NSGA-II and MOMA have a similar relationship as in the toy dynamic problem, although their average I_C values are shown to be significantly different given the degrees of freedom. NSGA-II has the higher average value, but it is not proven to be better than MOMA for the full problem.

Fig. B.21 presents the box-and-whisker plots of the binary-ϵ I_ϵ indicators for the three comparisons. GAMMA-PC has lower I_ϵ values than NSGA-II or MOMA, but its boxes do not hug the 1.0 line here as it did in the toy dynamic problem. Table B.17 shows the statistical analysis of Fig. B.21. It shows that the samples for

Table B.16: Statistical evaluation if GAMMA-PC performs better than NSGA-II or MOMA according to the $I_C(A, B)$ metric. First, the Wilcoxon-rank sum test is used to determine if the samples belong to the same distribution, with the null hypothesis $H_{wr,0} : I_C(A, B) = I_C(B, A)$ and the alternative hypothesis $H_{wr,1} : I_C(A, B) \neq I_C(B, A)$. Then, if $H_{wr,1}$ is accepted, the student-t test is applied to determine if the difference between the samples corresponds to the null hypothesis $H_{st,0} : I_C(A, B) - I_C(B, A) \geq 1.0 - I_C(B, A)$ or to the alternative hypothesis $H_{st,1} : I_C(A, B) - I_C(B, A) < 1.0 - I_C(B, A)$. The degrees of freedom was 798 for each test, and the significance level for each test was set at 0.016 to ensure an overall Type I error rate of 5% for both comparisons.

	Wilcoxon-Rank Sum Test	Student-t Difference Test
A = NSGA-II, B = GAMMA-PC		
test statistic	-24.2	-210.7
p-value	< 0.00001	< 0.00001
Accepted Hypothesis	H_1	H_1
A = MOMA, B = GAMMA-PC		
test statistic	-24.4	-184.3
p-value	< 0.00001	< 0.00001
Accepted Hypothesis	H_1	H_1
A = NSGA-II, B = MOMA		
test statistic	6.25	-55.7
p-value	< 0.00001	< 0.00001
Accepted Hypothesis	H_1	H_1

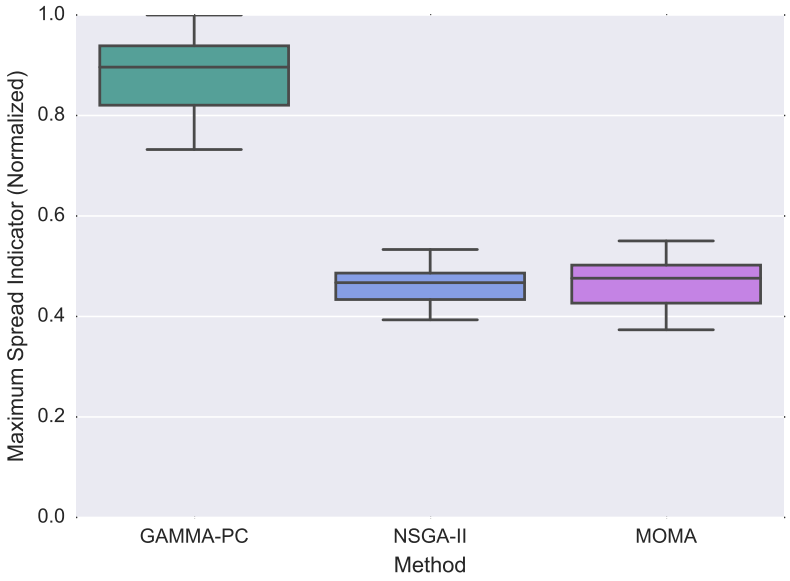


Figure B.22: Box plot of the maximum spread values for each algorithm over 20 experimental runs.

GAMMA-PC are statistically different than its counterparts, but it does not support the hypothesis that GAMMA-PC performs better by this metric. MOMA is shown to have a lower average I_ϵ value than NSGA-II in Fig. B.21(c), but it also cannot be proven to perform better.

The relationships of the alorithms with regards to diversity are similar to what was shown for the toy dynamic problem. The maximum spread indicators for GAMMA-PC, NSGA-II, and MOMA in the full dynamic problem are shown in Fig. B.22. The difference in the diversity is statistically signficant (refer to Appendix). As in the toy problem, GAMMA-PC produces solutions that are approximately twice as diverse as NSGA-II or MOMA. The diversity of NSGA-II and MOMA are also similar.

In the full dynamic problem, GAMMA-PC produced the same level of diversity but lower quality approximation sets than in the toy problem. Figs. B.23, B.24,

Table B.17: Statistical evaluation if GAMMA-PC performs better than NSGA-II or MOMA according to the $I_\epsilon(A, B)$ metric. First, the Wilcoxon-rank sum test is used to determine if the samples belong to the same distribution, with the null hypothesis $H_{wr,0} : I_\epsilon(A, B) = I_\epsilon(B, A)$ and the alternative hypothesis $H_{wr,1} : I_\epsilon(A, B) \neq I_\epsilon(B, A)$. Then, if $H_{wr,1}$ is accepted, the student-t test is applied to determine if the difference between the samples corresponds to the null hypothesis $H_{st,0} : I_\epsilon(B, A) - I_\epsilon(A, B) \geq I_\epsilon(B, A) - 1.0$ or to the alternative hypothesis $H_{st,1} : I_\epsilon(B, A) - I_\epsilon(A, B) < I_\epsilon(B, A) - 1.0$. The degrees of freedom was 798 for each test, and the significance level for each test was set at 0.016 to ensure an overall Type I error rate of 5% for both comparisons.

	Wilcoxon-Rank Sum Test	Student-t Difference Test
A = GAMMA-PC, B = NSGA-II		
test statistic	-24.5	-9.35
p-value	< 0.00001	< 0.00001
Accepted Hypothesis	H_1	H_1
A = GAMMA-PC, B = MOMA		
test statistic	-24.5	-11.3
p-value	< 0.00001	< 0.00001
Accepted Hypothesis	H_1	H_1
A = MOMA, B = NSGA-II		
test statistic	17.5	-13.4
p-value	< 0.00001	< 0.00001
Accepted Hypothesis	H_1	H_1

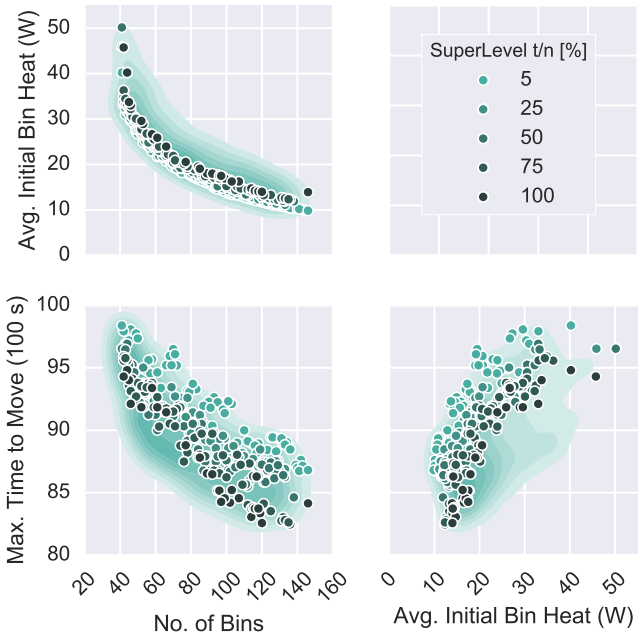


Figure B.23: Two-dimensional scatter matrix of the 3-dimensional empirical attainment function produced by GAMMA-PC for the full dynamic problem.

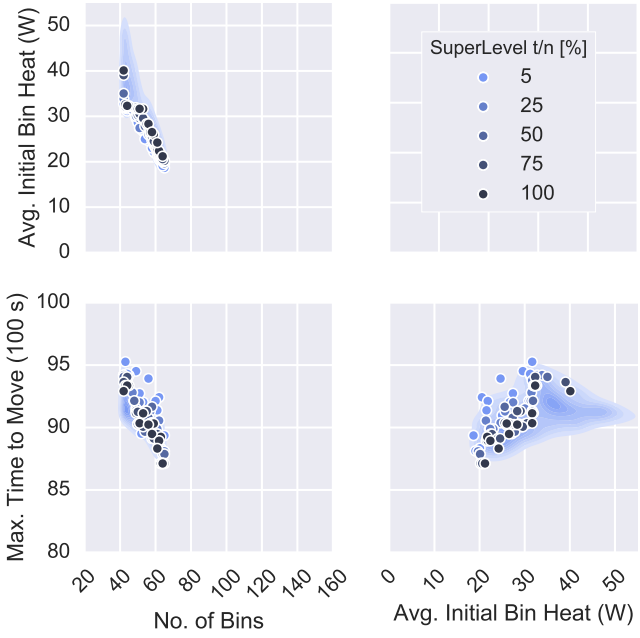


Figure B.24: Two-dimensional scatter matrix of the 3-dimensional empirical attainment function produced by NSGA-II for the full dynamic problem.

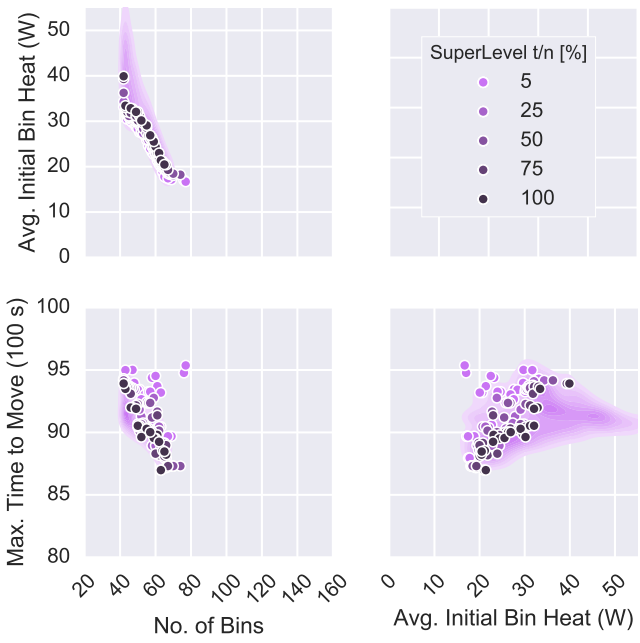


Figure B.25: Two-dimensional scatter matrix of the 3-dimensional empirical attainment function produced by MOMA for the full dynamic problem.

and B.25 present the partial two-dimensional scatter matrices of the empirical attainment functions for GAMMA-PC, NSGA-II, and MOMA, respectively. Once again, GAMMA-PC has explored the objective space at a much higher level than NSGA-II or MOMA. However, at the very low end of the x-axis (No. of Bins), GAMMA-PC produced solutions with higher levels of initial bin heat and larger maximum time values. Above about 60 bins, the initial heat levels of the GAMMA-PC solutions fall in the same range as those from NSGA-II and MOMA, but the maximum time values are still larger. GAMMA-PC did not demonstrate better performance in the full dynamic problem because either its exploitation ability was weaker or the underlying structure of the Pareto front was too complex to both explore and exploit solutions.

B.6 Conclusions

In this paper, we have introduced a new dynamic bin packing problem for storing cooling objects and a metaheuristic designed to work well in its mixed-variable environment named GAMMA-PC. The dynamic bin packing problem was discussed with the idea of cookie production at a bakery as a vehicle for explanation. The cookies arrived in batches at a cooling rack with a limited capacity and were packed into boxes with the competing goals of minimizing the number of boxes used, the initial heat of each box, and the maximum time until the boxes could be moved to the storefront. While a baker probably would not sell partially-full boxes to customers, the problem is able to represent more complex real-world applications, such as loading dry cask canisters with used nuclear fuel. To the best of our knowledge, this paper is the first to formulate a dynamic bin packing problem in this manner.

GAMMA-PC was applied to twenty standard bin packing problems and to a small and large version of the dynamic bin packing problem. GAMMA-PC performed as well as NSGA-II, MOMA, MOMAD, and MOEPSO, and in some cases better than NSGA-II, MOMAD, and MOEPSO, in the standard problems. It was also better at exploring the objective space in the dynamic problems. However, while it performed better overall in the small version of the dynamic problem, its performance was not proven to be better or worse than that of NSGA-II or MOMA in the large dynamic problem. The graphs of the empirical attainment functions suggest that the increase in problem size exacerbated the weakness hinted at in the low end of the x-axis in the static and toy problems. This suggests that GAMMA-PC may have a weak exploitative ability or that the exploration has been overemphasized.

Future research will need to find a better balance between exploration and exploitation for GAMMA-PC to be helpful with larger packing problems. One sug-

gestion would be to implement a more rigorous procedure for updating the operator probabilities. Instead of basing new local search probablities on the size of the local neighborhood around solutions in the approximation set with ideal values, the update procedure could be based on the evolving performance of each local search operator. An example of this would be the "compass" mechanism discussed in [174], which would consider how each operator makes improvements in fitness values and diversity as the calculation evolves. This mechanism could also be integrated with the selection of the crossover operator. While it may use more memory during the calculation, it is a promising avenue for better exploration and exploitation control.

B.7 Appendix

In [175], the authors advocate the use of the Kruskal-Wallis test to evaluate unary quality indicators with three or more independent samples. The unary indicators investigated here are the maximum spread indicators, the Pareto front absolute effiency, and the distance measures. Therefore, the Kruskal-Wallis test was applied to the samples to determine the presence of a signifcant difference before applying the student-t difference tests. The degrees of freedom for the student-t tests were found using the Satterthwaite approximation given in (B.43) [176].

$$df = \frac{\left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}\right)^2}{\frac{(S_1^2/n_1)^2}{n_1-1} + \frac{(S_2^2/n_2)^2}{n_2-1}} \tag{B.43}$$

Table B.18 presents the statistical evaluation of the maximum spread indicator for the full dynamic problem. The Kruskal-Wallis test showed that there were statistically signficant differences in the maximum spread indicators for GAMMA-PC, NSGA-II, MOMA, MOMAD, and MOEPSO ($\chi^2(2) = 83.3$, $p < 0.00001$). Based on this evidence, student-t tests were applied to the differences in the spread values be-

Table B.18: Statistical evaluation if GAMMA-PC produces more diverse solutions than NSGA-II, MOMA, MOMAD, or MOEPSO by the maximum spread indicator for the static problem. Each comparison was made using the student t-test for the null hypothesis $H_0 : D(\text{GAMMA-PC}) \leq D(B)$ and the alternative hypothesis $H_1 : D(\text{GAMMA-PC}) > D(B)$. The degrees of freedom were determined using the Satterthwaite method, and the significance level for each test was set at 0.0125 to ensure an overall Type I error rate of 5% for both comparisons.

$B =$	NSGA-II	MOMA	MOMAD	MOEPSO
df	20	38	20	21
t-statistic	19.0	2.55	19.1	16.9
p-value	<0.00001	0.0075	<0.00001	<0.00001
Accepted Hypothesis	H_1	H_1	H_1	H_1

tween GAMMA-PC and the others. The alternative hypothesis was accepted for each test, indicating that GAMMA-PC produces more diverse solutions at a statistically significant level.

Table B.19 presents the statistical evaluation of the box plots shown in Figs. B.11 and B.12. The Kruskal-Wallis test showed that there were statistically significant differences in the absolute efficiencies ($\chi^2(2) = 7.89$, $p = 0.0194$), the average distances ($\chi^2(2) = 9.62$, $p = 0.0081$), and the maximum distances ($\chi^2(2) = 8.18$, $p = 0.0167$) for GAMMA-PC, NSGA-II, and MOMA. While the absolute efficiency of GAMMA-PC cannot be proven to be different, the difference between its distance and the other two algorithms' is statistically significant.

Table B.20 presents the statistical evaluation of the maximum spread indicator for the toy dynamic problem. The Kruskal-Wallis test showed that there were statistically significant differences in the maximum spread indicators for GAMMA-PC, NSGA-II, and MOMA ($\chi^2(2) = 12.5$, $p = 0.0019$). Based on this evidence, student-t

Table B.19: Statistical evaluation if GAMMA-PC performs better than NSGA-II or MOMA by the Pareto front measures. Each comparison was made using the student-t test for the null hypothesis H_0 : $value$(GAMMA-PC) $\leq$ $value(B)$ ($\geq$ for absolute efficiency) and the alternative hypothesis H_1 : $value$(GAMMA-PC) $>$ $value(B)$ ($<$ for absolute efficiency). The degrees of freedom were determined using the Satterthwaite method, and the significance level for each test was set at 0.025 to ensure an overall Type I error rate of 5% for both comparisons.

	Absolute Efficiency	Avg. Distance	Max. Distance
B = NSGA-II			
df	–	5	7
statistic	–	6.57	3.57
p-value	–	0.0006	0.0045
Accepted Hypothesis	–	H_1	H_1
B = MOMA			
df	4	5	7
statistic	2.18	6.52	3.11
p-value	0.0474	0.0006	0.0085
Accepted Hypothesis	H_0	H_1	H_1

Table B.20: Statistical evaluation if GAMMA-PC produces more diverse solutions than NSGA-II or MOMA by the maximum spread indicator for the toy dynamic problem. Each comparison was made using the student t-test for the null hypothesis $H_0 : D(\text{GAMMA-PC}) \leq D(B)$ and the alternative hypothesis $H_1 : D(\text{GAMMA-PC}) > D(B)$. The degrees of freedom were determined using the Satterthwaite method, and the significance level for each test was set at 0.025 to ensure an overall Type I error rate of 5% for both comparisons.

$B =$	NSGA-II	MOMA
df	5	6
t-statistic	16.4	13.5
p-value	<0.00001	<0.00001
Accepted Hypothesis	H_1	H_1

tests were applied to the differences between the spread values for GAMMA-PC and those for NSGA-II and MOMA. Table B.20 shows the results of these two tests, indicating that GAMMA-PC produces more diverse solutions than NSGA-II or MOMA.

Table B.21 presents the statistical evaluation of the maximum spread indicator for the full dynamic problem. The Kruskal-Wallis test showed that there were statistically signficant differences in the maximum spread indicators for GAMMA-PC, NSGA-II, and MOMA ($\chi^2(2) = 39.4$, $p < 0.00001$). Based on this evidence, student-t tests were applied to the differences in the spread values between GAMMA-PC and NSGA-II and between GAMMA-PC and MOMA. The alternative hypothesis was accepted for each test, indicating that GAMMA-PC produces more diverse solutions at a statistically significant level.

Table B.21: Statistical evaluation if GAMMA-PC produces more diverse solutions than NSGA-II or MOMA by the maximum spread indicator for the full dynamic problem. Each comparison was made using the student t-test for the null hypothesis $H_0 : D(\text{GAMMA-PC}) \leq D(B)$ and the alternative hypothesis $H_1 : D(\text{GAMMA-PC}) > D(B)$. The degrees of freedom were determined using the Satterthwaite method, and the significance level for each test was set at 0.025 to ensure an overall Type I error rate of 5% for both comparisons.

$B =$	NSGA-II	MOMA
df	28	30
t-statistic	19.99	19.19
p-value	<0.00001	<0.00001
Accepted Hypothesis	H_1	H_1