Development of a Greedy Method for Used Fuel Selection in Dry Cask Storage

Kristina Yancey Spencer, Pavel V. Tsvetkov, Joshua J. Jarrell

Texas A&M University, 3133 TAMU, College Station, TX 77843-3133, USA
Oak Ridge National Laboratory, TN, USA

INTRODUCTION

In 2006, the IAEA recommended that nuclear utilities reevaluate the common practice of loading the oldest and coldest used fuel assemblies from spent fuel pools into dry cask storage [1]. Since then relatively little research has focused on the dry cask loading problem. With the development of the Used Nuclear Fuel-Storage, Transportation & Disposal Analysis Resource and Data System (UNF-ST&DARDS) [2], a database containing the most accurate information about U.S. used fuel available, a more nuanced approach is possible. Recently, an optimization tool for UNF-ST&DARDS was developed that explores this recommendation.

A new metaheuristic algorithm was developed in this research to handle the complexities of the dry cask loading problem. It incorporates greedy randomized adaptive search procedure (GRASP) [3] heuristics into an evolutionary framework based on the nondominated sorting genetic algorithm II [4] and features partial decomposition of the objective space to group similar solutions together during crossover operations. It was named the GRASP-enabled adaptive multiobjective memetic algorithm with partial clustering, or GAMMA-PC.

THE DRY CASK LOADING PROBLEM

The dry cask loading problem belongs to the general category of bin packing problems, a type of NP-hard combinatorial problem [5]. The main objective of bin packing problems is to fit a number of items N into as few bins as possible within constraints such as weight or size limits. Applied to the selection of used fuel assemblies, this goal is incorporated with the competing objectives of minimizing the average initial heat in each bin (i.e. cask) and reducing the time required for all of the bins to meet transportation regulations.

The mathematical paradigm of the dry cask loading problem uses the standard multiobjective problem format, given in (1).

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

In (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 best feasible solutions involve trade-offs among the competing objectives. Comparisons between solutions are made on the basis of domination, where a solution with objective vector $u = (u_1, ..., u_m)^T$ **dominates** another with vector $v = (v_1, ..., v_m)^T$ iff $\forall\theta \in \{1, ..., m\}$, $u_\theta \leq v_\theta$ and $u \neq v$. A feasible solution is optimal if it is not dominated by any other solution.

To optimize the selection of used fuel assemblies, it is necessary to consider how many casks to use, which assemblies to place into each cask, and when to perform transfer procedures. Equation 2 combines these variables into the general form of the decision vector $\mathbf{x}$.

$$\mathbf{x} = \begin{bmatrix} x & y & t_{fill} \end{bmatrix} \tag{2}$$

This combination expands on the standard bin packing problem decision vector format with the inclusion of the continuous time variable t_{fill}, an $\overline{M} \times 1$ array denoting the time each cask is loaded with used fuel assemblies. The constant $\overline{M}$ represents the theoretical maximum number of bins. The decision variables x and y are binary packing matrices of size $\overline{M} \times N$ and $\overline{M} \times 1$ respectively. The loading matrix x identifies which assemblies are located in each cask, and the bin array y indicates which casks are scheduled for loading.

Loading used nuclear fuel into dry cask storage is an intensive process, so ideally, every cask would be filled to capacity. However, 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, evaluated in Eq. 3.

$$\overline{M} = \left\lceil \frac{N}{0.75 C_{cask}} \right\rceil \tag{3}$$

Here, the variable C_{cask} represents the capacity of one canister.

The combined tri-objective vector is formulated mathematically in Eq. 4.

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} c \\ \frac{1}{c} \sum_{i=1}^{c} w_{r,i} Q_i \\ max(t_{trans,min,i}, i = \{1, ..., c\}) \end{pmatrix} \tag{4}$$

In $\mathbf{F}(\mathbf{x})$, the variable c represents the number of casks in use and can be found through a summation of the binary array y. The second element represents the average initial heat of the bins where $w_{r,i}$ is a weighting factor for the initial heat in cask i, Q_i (kW). The last element represents the maximum time required for casks to be ready for transportation. The variable $t_{trans,min,i}$ represents the earliest date that cask i would meet transportation dose limits and is found using the modified regula falsi method.

The search space for new solutions to Eq. 4 is constrained to meet regulatory and user-defined requirements. The feasible region $\mathbf{\Omega}$ can be seen as the intersection of subregions defined by these constraints, as shown in Eq. 5.

$$x \in \mathbf{\Omega} = \Omega_{bpp} \cap \Omega_{lc} \cap \Omega_{pool} \cap \Omega_{oper} \tag{5}$$

The subregions represent 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 independent spent fuel storage installation (ISFSI).

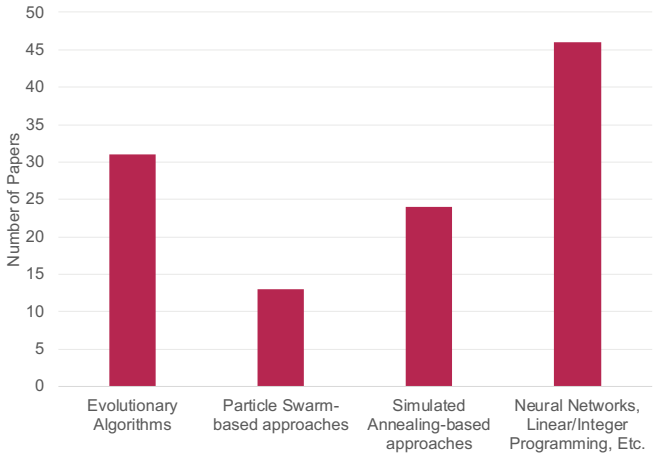

Fig. 1. Distribution of optimization methodologies used for combinatorial problems in nuclear engineering. This bar chart was created based on a survey of 100 papers.

OPTIMIZATION OF COMBINATORIAL PROBLEMS

Existing dry cask loading studies are limited. During the literature review, only four programs were identified. The two professional software programs, Studsvik's MARLA [6] or the Electric Power Research Institute's CASKLOADER [7], help utilities pick assemblies that meet the CoC requirements, but neither disclose the method used to perform their optimization. The two research studies on canister loading do describe their method, although neither uses a state-of-the-art optimization methodology that would enable efficient computation in such a large search space [8, 9]. Moreover, all of the programs only select used fuel from the current pool inventory.

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, although other areas such as fuel cycle optimization and maintenance planning 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. Fig. 1 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 methods in Fig. 1 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 [10]. Simulated annealing can vary between these extremes based on a set of sensitive parameter settings [11].

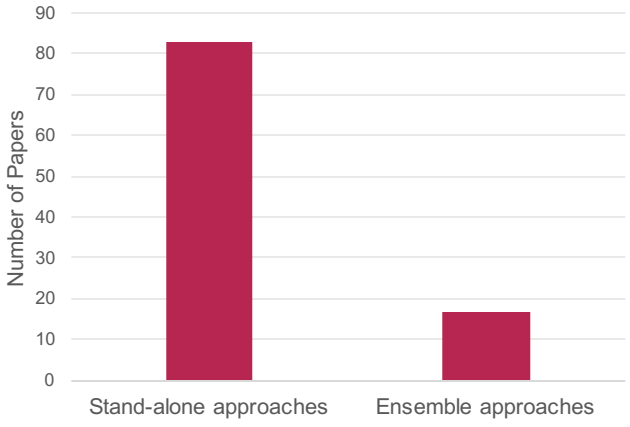

Fig. 2. Comparison of stand-alone versus ensemble optimization approaches in nuclear engineering. This bar chart was created based on a survey of 100 papers.

It is possible to combine the advantages of different algorithms by utilizing **ensemble learning** in which decision-making strategies or learning algorithms work together to improve performance [12]. Fig. 2 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.

Beyond the nuclear field, multiobjective evolutionary algorithms (MOEAs) have been the most popular method for solving multiobjective optimization problems [13]. 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.

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 [13]. 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 [14]. 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.

GAMMA-PC

Three populations are maintained throughout GAMMA-PC: P, Q, and P_{EA}. The solutions in P represent the "parent" solutions for a generation, and the solutions in Q are solutions produced by genetic operators and found through 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 *GRASP-DBPP*,

2. While the stopping condition is not satisfied, do:
3. Select set P using NSGA-II-styled binary selection,
4. Update P_{EA},
5. Update Q using *Crossover-PC*,
6. Update Q by *Mixed-Variable-Genetic-Mutation*,
7. Transform new solutions in Q using *GRASP-DBPP*,
8. Add solutions to Q by *Adaptive-Local-Search*,
9. Every g generations, *Truncate-External-Archive*,
10. *Update-Operator-Probabilities*,
11. End while, and return P_{EA}.

The algorithm is initialized by generating a set of solutions using packing heuristics designed for the dynamic bin packing problem (DBPP) embedded within the GRASP procedure. This operator *GRASP-DBPP* constructs new solutions, compares each with a "neighbor" solution, and returns the best solution to be added to the set Q.

The *Construct-New-Solution* module within *GRASP-DBPP* can build a new solution either from scratch or given a permutation of the item indices, i.e. a real-valued chromosome representation of a solution. It uses three different packing heuristics: the least loaded approach to distribute the items evenly among the bins, the first fit approach to reduce the number of bins needed, and the dot product strategy discussed in [5] to balance the heat load and available capacity. When a new bin is opened, a continuous greedy function to select the fill time is employed. This contribution combines the idea of Monte Carlo selection with that of a restricted candidate list. The function keeps track of the available space in the spent fuel pool as a function of time. When called, it restricts the timeline and returns a randomly selected time from within that range.

At the beginning of each generation, a new set P is assembled from the best solutions in set Q, and any new nondominated solutions are added to P_{EA}. The solutions are then sent to the *Crossover-PC* operator. The *Crossover-PC* operator sends a fraction f_{ran} of P straight to the general crossover operator. Then, it carefully sorts the remaining solutions into N_c clusters before sending each cluster to the crossover operator. The clusters are grouped together using the Tchebycheff approach [15]. The value for N_c used in this research was 6, or $2m$. The mutation rate is increased by a small amount for any empty cluster found. All of the solutions produced by the multiple crossovers are combined to form set Q.

Following the *Crossover-PC* operator, random solutions in Q undergo genetic mutation. The genetic operations in both Steps 5 and 6 are modified from their traditional form to account for the mixed-variable environment. Both handle the chromosome representation of the packing and the t_{fill} decision variable separately. The *Mixed-Variable-Genetic-Crossover* operator performs single-point crossover on each, while the *Mixed-Variable-Genetic-Mutation* operator performs a two-point swap in the chromosome representation only. Then, the $\mathbf{t}_{fill}$ array is modified using Eq. 6.

$$t_{fill,i,g+1} = t_{fill,i,g} + \mathcal{N}(0, \sigma_{cat}),\ \forall\ i \in \{1, ..., \overline{M}\} \qquad (6)$$

Here, g represents the generation, and σ_{cat} is the standard deviation of the values for $t_{fill,i}$ within a particular time range. The function $\mathcal{N}(0, \sigma_{cat})$ returns a randomly generated value from within the Gaussian defined by σ_{cat}. Any new chromosome-t_{fill} pair found during genetic operations is reconstructed before being added to set Q.

In Step 8 of GAMMA-PC, local search is performed on the new solutions in Q. 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.

The local search probabilities are 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. After every g generations, P_{EA} is truncated to keep the size of the archive below a preset level.

RESULTS

To demonstrate its performance, GAMMA-PC was applied to ISFSIs at Vermont Yankee, Comanche Peak, and Zion nuclear power plants. Its performance was evaluated through comparisons to test solutions representing the oldest and coldest (OC) loading strategy.

Figure 3 presents results from one of these scenarios. It shows 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.

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. Another finding was that the third objective function was sensitive to assembly-level characteristics and often reduced to a single number independent of the other two objective function values. Finally, the analysis of the individ-

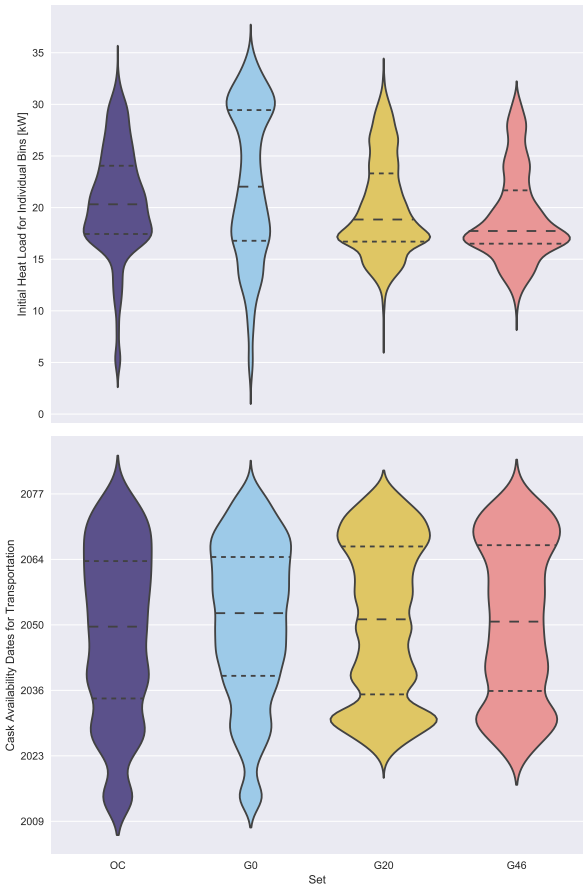

Fig. 3. Violin plots comparing the cask characteristics of solutions for the Comanche Peak dry cask loading problem. The OC solutions are compared to solutions found by GAMMA-PC during initialization, in generation 20, and in the final set (G46). The top plot shows the distributions of initial cask heat loads, and the bottoms shows the eligibility dates for transportation.

ual cask characteristics highlighted the fact that the loading problem paradigm does not directly address balance among the casks. Future work should incorporate this as a goal.

ACKNOWLEDGEMENTS

Research supported 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).

To see the list of articles included in the survery of 100 combinatorial nuclear studies, please visit https://github.com/kyspencer/Survey-on-Nuclear-Combinatorial-Optimization-Studies.

REFERENCES

1. IAEA, "Optimization Strategies for Cask Design and Container Loading in Long Term Spent Fuel Storage," Report IAEA-TECDOC-1523, International Atomic Energy Agency (2006).
2. K. BANERJEE, J. SCAGLIONE, and R. LEFEBVRE, "Integrated Data and Analysis Tool for Used Nuclear Fuel Management," in "Transactions of the American Nuclear Society," American Nuclear Society, American Nuclear Society (November 2014), vol. 111, pp. 338 – 341.
3. T. A. FEO and M. G. RESENDE, "Greedy Randomized Adaptive Search Procedures," *Journal of Global Optimization*, **6**, 109–133 (1995).
4. K. DEB, A. PRATAP, S. AGARWAL, and T. MEYARIVAN, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, **6**, *2*, 182 – 197 (2002).
5. N. DAHMANI, F. CLAUTIAUX, S. KRICHEN, and E.-G. TALBI, "Self-adaptive metaheuristics for solving a multiobjective 2-dimensional vector packing problem," *Applied Soft Computing*, **16**, 124 – 136 (2014).
6. D. KNOTT and C. OYARZUN, "Description of the shuffle design software MARLA," *Progress in Nucl. Energy*, **53**, *6*, 607–617 (2011).
7. EPRI, "2014 Research Portfolio: Used Fuel and High-Level Waste Management (QA)," Tech. rep., Electric Power Research Institute (2014).
8. G. ŽEROVNIK, L. SNOJ, and M. RAVNIK, "Optimization of spent nuclear fuel filling in canisters for deep repository," *Nucl. Sci. and Eng.*, **163**, *2*, 183–190 (2009).
9. 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*, **71**, *0*, 125–129 (2014).
10. 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*, **50**, *0*, 117–125 (2012).
11. R. HAYS and P. J. TURINSKY, "BWR in-core fuel management optimization using parallel simulated annealing in FORMOSA-B," *Progress in Nucl. Energy*, **53**, *6*, 600–606 (2011).
12. S. SOARES, C. H. ANTUNES, and R. ARAÚJO, "Comparison of a genetic algorithm and simulated annealing for automatic neural network ensemble development," *Neurocomputing*, **121**, *0*, 498–511 (2013).
13. A. ZHOU, B.-Y. QU, H. LI, S.-Z. ZHAO, P. N. SUGANTHAN, and Q. ZHANG, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evol. Computation*, **1**, *1*, 32 – 49 (2011).
14. 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.*, **171**, 27 – 49 (2009).
15. Q. ZHANG, W. LIU, E. TSANG, and B. VIRGINAS, "Expensive Multiobjective Optimization by MOEA/D with Gaussian Process Model," *IEEE Transactions on Evolutionary Computation*, **14**, *3*, 456–473 (2010).