Proceedings Paper:
Webster, A., Jia, X. and Xie, S.Q. (2024) Cost-Optimal Cutting and Packing for Nuclear Decommissioning. In: 2024 30th International Conference on Mechatronics and Machine Vision in Practice (M2VIP). 2024 30th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), 03-05 Oct 2024, Leeds, United Kingdom. Institute of Electrical and Electronics Engineers (IEEE). ISSN: 2996-4156. EISSN: 2996-4164.

https://doi.org/10.1109/m2vip62491.2024.10746009

Cost-Optimal Cutting and Packing for Nuclear Decommissioning

Aron Webster*, Xiaodong Jia†, Sheng Quan Xie‡
*†*School of Chemical and Process Engineering* ‡*School of Electronic and Electrical Engineering*
University of Leeds
Leeds, UK
{*pmaow, †X.Jia, ‡S.Q.Xie}@leeds.ac.uk

***Abstract*—During nuclear decommissioning, large structures often need to be cut into smaller parts before being packed into containers. Cutting and packing incur costs, and improving efficiency in one often increases costs in the other, requiring a trade-off. Considering the operational costs of different cutting tools and container types, determining the optimal number of parts to cut a structure into, and how to pack these parts efficiently, is challenging. This paper presents a novel voxel-based cutting and packing optimization algorithm designed to minimize overall costs for decommissioning nuclear structures. We define cost as comprising two factors: cutting cost, which depends on the amount of cutting required and the type of tool used, and packing cost, which is determined by the number of containers needed and the unit cost of a container. We demonstrate that our algorithm can successfully minimize overall cutting and packing costs. Additionally, we show how our algorithm can be used to compare different decommissioning scenarios by cutting a simple structure found in nuclear decommissioning and packing the pieces into two different container types with different unit costs. Our results also show that the algorithm outperforms a minimal cutting approach, where a minimum number of cuts is used to segment the structure.**

***Index Terms*—Nuclear decommissioning, Cutting and packing, Cost optimisation, Multi-container packing, Genetic algorithm**

I. Introduction

At some stage in the decommissioning of nuclear installations, it is inevitable that large metal structures (e.g., reactor vessels, glove boxes, pipe structures) will require size reduction via cutting before they can be packed into containers for temporary storage, permanent disposal or simply for transportation. Both cutting and packing cost money and usually, efficiency or cost-saving in one is achieved at the expense of the other (meaning a trade-off is required).

Consider the structure depicted in Fig.1. The left case uses few cuts resulting in low cutting cost, but requires two containers, increasing packing cost. The right case has many cuts leading to high cutting cost, but all parts fit into one container, reducing packing costs. When considering the total cost of cutting and packing, it is difficult to know which approach is more cost effective prior to performing the cutting and packing. Furthermore, it may be possible to cut the structure with fewer cuts than in the right case and still pack the pieces into a single container, potentially yielding a more cost-efficient solution than either displayed approach.

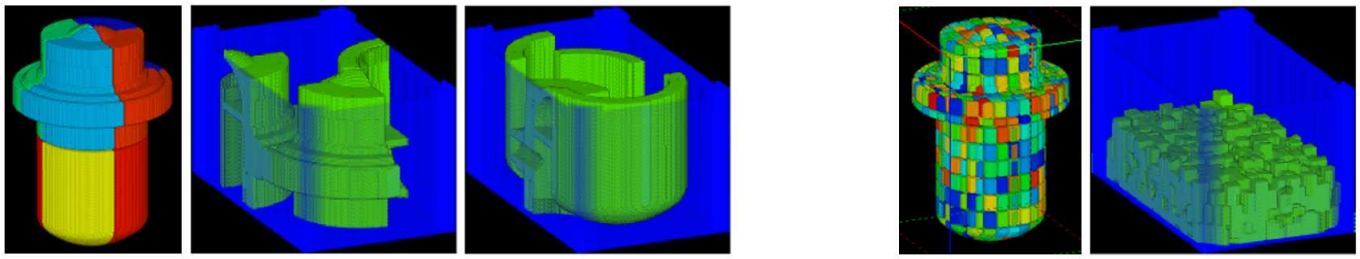
This paper is supported by UK Research and Innovation [EP/S022295/1].

The problem with existing cutting optimization [2], [3] and packing optimization [4], [5] algorithms for nuclear decommissioning are that they fail to address the trade-off between cutting and packing. The algorithms in [2], [3] focus solely on minimizing the amount of cutting required while ensuring all parts are within the maximum allowed size, without considering packing. Conversely, algorithms in [4], [5] only optimize packing for a fixed set of objects without accounting for the cutting process.

To date, there are very few examples in literature where this trade-off between cutting and packing has been considered, with all current examples being developed for 3D printing [6]–[9]. The goal of these algorithms is to partition an input object (which may be larger than the printing volume) and then pack the pieces into the printing volume with the goal being to minimise: 1.) the printing time/support material, 2.) the space required to pack the parts (for storage/shipping) and 3.) the number of parts for ease of assembly.

The problem with approaches [6]–[8] are their heavy reliance on a good initial partitioning. In [6], cuts can only be altered to improve packing but not added or removed, in [7] cuts can only be removed (but not added or altered), and in [8] cuts can only be added (but not removed or altered). This problem could be alleviated by re-running the algorithm with different initial decompositions, but none of them do this. Furthermore, with approaches [6] and [8], they only allow cuts to be modified/added to locally improve the current packing structure. Each time cuts are altered, the objects to pack change. By failing to run packing optimisation from the start each time the objects change, the algorithm may miss good packing solutions. The algorithm presented in [9] doesn't suffer from these problems, however its main limitation lies in the fact that it requires the user to specify a target packing efficiency and maximum number of cut parts (with the quality of solutions being sensitive to these parameters) and may potentially get stuck if the part limit is reached before the target packing efficiency is reached.

The final major limitation with approaches [6]–[9] is that they only pack objects into a single container, whereas nuclear decommissioning often requires multiple containers. Any algorithm developed for nuclear applications should have the ability to pack objects across multiple containers whilst seeking to minimise the amount of containers required.



(a) (b)

Fig. 1. Example of two ways to cut and pack a structure. (a) Solution with few cuts (low cutting cost) and more containers (high packing cost). (b) Solution with many cuts (high cutting cost) and fewer containers (low packing cost). Images from [1].

In this paper, we present a novel voxel-based cutting and packing algorithm to optimally partition an input structure and pack the cut parts across multiple containers. We allow the algorithm to modify existing cuts as well as add and remove them, and we run packing optimisation each time a new cutting pattern is generated. We also represent cutting and packing cost in a way which allows the user to set cost weights depending on the cost of the cutting tool and container type being used. This work presents a first attempt at applying cutting and packing optimisation (in a linked fashion) to nuclear decommissioning.

II. Problem Description

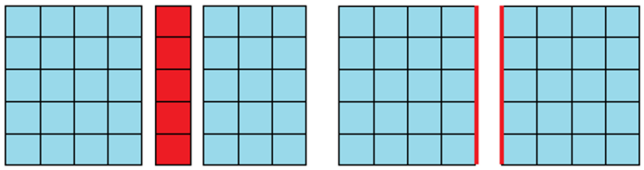
In our algorithm, we convert input mesh models into voxels (3D equivalent of pixels representing small cubic units), with the resolution of the input structure and container set by the user.

For a voxelised 3D model of a structure, the goal of our optimisation algorithm is to find a way to cut and pack the structure such that the sum of the cutting and packing costs (total cost) is minimised. We define the cutting cost as $w_1 V_c$, where V_c is the intersection volume between the cuts and the structure and w_1 is the cost per unit volume for cutting (set by the user depending on the type of cutting tool used). Since we allow the user to define the width of the cuts, if the user selects zero-width, the intersection volume is instead calculated as the intersection area between the cuts and the structure (see Fig. 2). We define the packing cost as $w_2 N_p$, where N_p is the number of containers required to pack all the cut parts and w_2 is the cost per container (set by the user depending on the type of container used). Thus we express the optimisation objective as the desire to minimise:

$$T_{cost} = w_1 V_c + w_2 N_p \tag{1}$$

Subject to the following constraints:

1) All cut parts must be small enough to fit into the containers.
2) There must be no overlap between packed parts and all packed parts must be within the container boundaries.
3) Each packing location must be accessible, with a collision-free trajectory from the top of the container to the desired packing location.



(a) (b)

Fig. 2. 2D example of a cut widths. (a) cut width = 1, red region is the intersection volume of the cutting plane and the object. (b) cut width = 0, red lines are the intersection area between the cutting plane and the object.

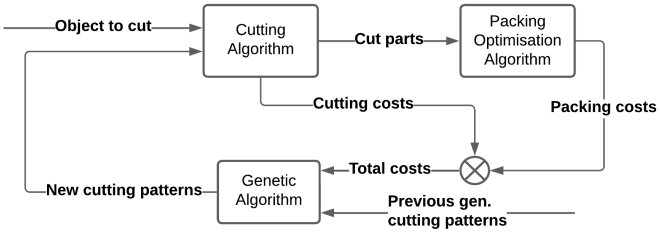
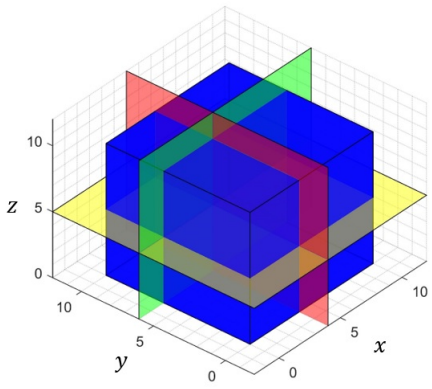


Fig. 3. Flowchart outlining the cutting and packing optimisation process.

III. Solution Approach

Our proposed method is to solve the optimisation problem in a feedback fashion using a Genetic Algorithm (GA) to generate cutting patterns which drive down total cost (see Fig. 3). The algorithm starts with a population of cutting patterns, where each chromosome is a list of cuts for a single cutting pattern and each gene is an individual cut. The cutting patterns are each applied to the structure to decompose it, and each cutting pattern's corresponding set of cut parts is then packed by the packing algorithm. The cutting and packing cost of each cutting/packing solution is then summed to get the total cost, which the algorithm uses to rank the cutting patterns from cheapest to most expensive. A new population of cutting patterns is then generated via crossover and mutation, and the process repeats again. With this approach, the packing algorithm is treated as part of the solution evaluation process, i.e. the outputs (costs) from both the cutting *and* packing algorithms are used to evaluate how good the cutting patterns are.



X	Y	Z
4	0	0
0	6	0
0	0	5

Fig. 4. Simple illustration of the planar cutting approach. Left - blue cube being cut with three planar cuts orthogonal to: x axis (red), y axis (green) and z axis (yellow). Right - cutting list with corresponding cuts colour coded.

A. Cutting Algorithm

We adopt a planar cutting approach where cuts are orthogonal to each axis and can pass through one another. Referring to Fig. 4 as an example, cut lists are implemented as a 3-column vector, where the column position of the non-zero value determines which axis the cut is orthogonal to, and the value itself dictates its location along the axis.

When performing crossover on two cut lists, we use single point crossover where both cut lists from both parents are split at random points. The top and bottom halves of the lists from both parents are then recombined to from two offspring. Mutation is then applied to both offspring. When performing mutation, the algorithm will cycle through the list of cuts and apply a small random change to a cut (with the chance of cut mutation dictated by the mutation rate) to move it slightly in the positive or negative direction.

To ensure all cut parts are small enough to be packed, we use a repair mechanism to add cuts if the distance between two successive cuts becomes too large. At the algorithms start, the user must set maximum x, y, z limits for the cut parts (which must be smaller than or equal to the container dimensions). The repair mechanism checks the distance between successive cuts along each axis. If the distance exceeds the cut-part limit for that axis, additional cuts are added by recursively dividing the distance until all sub-distances are within the limit. The repair mechanism also removes duplicate cuts (cuts on the same axis with the same values) and cuts that have been mutated outside the structures bounding box.

B. Packing Algorithm

To pack a cut set of parts into a minimal number of identical containers, we adapt a 2D packing approach from [10] called Partial Bin Packing (PBP).

PBP works by focusing on the optimal packing of one container at a time. It starts by opening an empty container and running an allocation algorithm to select a subset of cut parts to allocate to the container. The subset is then ordered by non-increasing volume and the parts are packed one by one, starting with the largest. If the algorithm finds that an object won't fit, the assignment is mended. During mending, the successfully packed parts are kept in the container and the object which failed to fit is placed into a temporary exclusion set. The allocation process is then re-run to select another subset from the unpacked items to try adding to the partially packed container. The subset is then packed, and if the algorithm encounters another object which cant fit, the repair process is repeated. This process of packing and repair will repeat until either all allocated objects are successfully packed, or there are no more objects left in the main object set to try adding to the container. Upon termination the packed container is closed, any objects in the temporary exclusion set are returned to the main object pool, a new container is opened, and the process repeats.

The allocation algorithm is a modified version of the knapsack problem where objects are selected to maximise the sum of their volumes without exceeding the available volume in the container [10]. The problem is expressed as:

$$Max. \sum_{k=1}^{n} \left(\frac{v_k}{v_{max}} \right)^2 q_k, \tag{2}$$

$$\sum_{k=1}^{n} v_k q_k \leq C_v, \quad 1 \leq k \leq n, \tag{3}$$

$$q_k \in \{0, 1\}, \quad 1 \leq k \leq n, \tag{4}$$

Where n is the number of unpacked objects in the main pool, v_k is the volume of object k, v_{max} is the largest object volume in n, C_v is the available volume in the container, and q_k is a binary variable taking the value of 1 when object k is allocated to the container and 0 otherwise.

The purpose of using the modified objective function in (2) as opposed to using the sum of object volumes directly, is to favour assigning larger objects. This prevents greedy (and poorer) solutions where small objects are packed together early on, leaving larger objects till the end [11].

Since we use a voxel representation for objects, the packing space is also discretised to form a lattice grid. To pack an object, the algorithm translates the object in discrete steps across the packing space lattice for all allowed orientations (with the angle increment set by the user). At each step, overlap detection is performed to check for collisions with previously packed objects or the container boundary. If there is no overlap, the location is stored as a feasible packing site. The object is then placed at the location that minimizes its height in the packing structure and checked for accessibility by translating it vertically to the top of the container, whilst checking for overlap. If no overlap occurs, the object is placed. If there is overlap, the next best site is tested. This process continues until an accessible site is found or the object cannot be packed.

IV. Implementation

The algorithm is implemented into the decommissioning software NuPlant™ [12], written in C++, and was run on a desktop PC with an Intel Xeon E5-1650 v3 3.50GHz processor

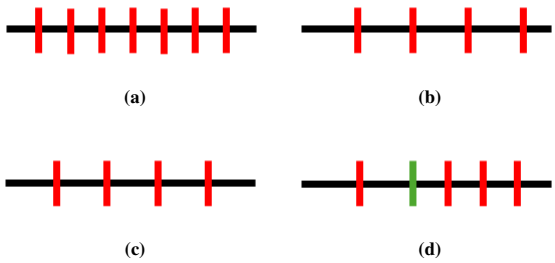


Fig. 5. Simple illustration of four minimal cutting approaches. Line length = 9, limit = 2, line origin (0) on left. (a) Recursive division - line segment recursively divided until distance between cuts ≤ 2. (b) Minimal cutting - number of cuts to add = $\lfloor 9/2 \rfloor$ = 4, cuts added at 2,4,6,8. (c) Even division - distance between cuts = 1.8. (d) Random single cut - random cut (green) added at 4, line segments either side recursively divided.

and 64Gb of RAM. The algorithm was run with a population size of 20 for 100 generations. We use binary tournament selection to select parents for crossover and mutation, and use elitism to retain the best solution in each generation. We generate new populations entirely via crossover, with mutation applied to each child solution. The mutation rate was set to 0.05. The allocation problem (equation 2) was also solved with a GA, running for 500 generations with a population size of 100.

To generate the initial population of cutting patterns, we initialise four cutting patterns with minimal cuts, and populate the rest with a random number of randomly placed cuts. For minimal cuts, we use four different approaches:

1) Recursive division - Each axis of the structures bounding box (x, y, z) is recursively split in half until each line segment is smaller than the limit for that axis (Fig. 5a).
2) Minimal cutting - For each bounding box axis, calculate minimum number of cuts as $n_c = \lfloor length/limit \rfloor$, then, for $i = 1, ..., n_c$, add a cut at $i * limit$ (Fig. 5b).
3) Even division - Divide each axis with evenly spaced cuts where the distance between cuts is smaller than its respective axis limit (Fig. 5c).
4) Random single cut - For each axis, place a single cut at a randomly generated point, then check the length of the line segment either side of the cut. If either one exceeds its respective axis limit, recursively divide the line segment (Fig. 5d).

For packing, we allow full object rotation in 90 degree increments about each axis giving 10 possible orientations (original orientation + 90, 180 and 270 degrees about each axis).

A. Test Cases

With our optimisation methodology, different cutting and packing scenarios can be considered based on information related to problem specific decommissioning protocols and costing models (which are often proprietary and will not be examined in this study). Instead we aim to show how our algorithm can offer flexibility relating to different cutting and packing requirements. More specifically, we will demonstrate how choice of different containers (with differing unit costs) will result in different total-cost cutting and packing solutions and how users can utilise this information when planning appropriate decommissioning strategies.

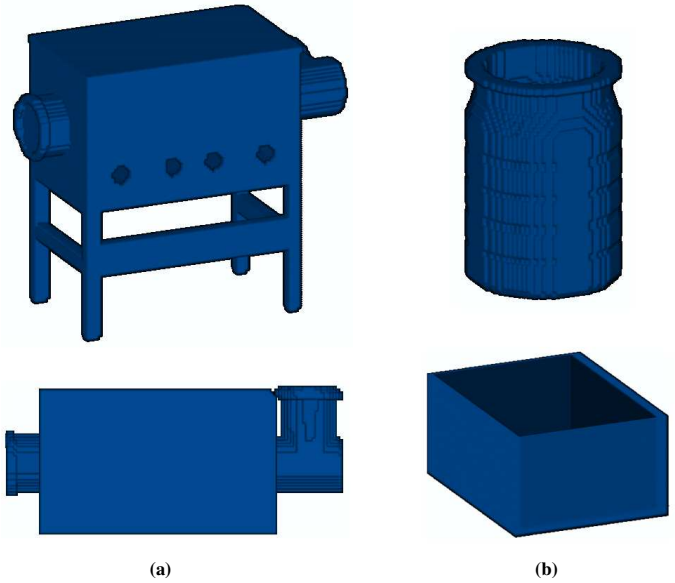


Fig. 6. Test structure and containers (voxelised). (a) Glove-box with top-down view. (b) Top - drum container, Bottom - cuboid container.

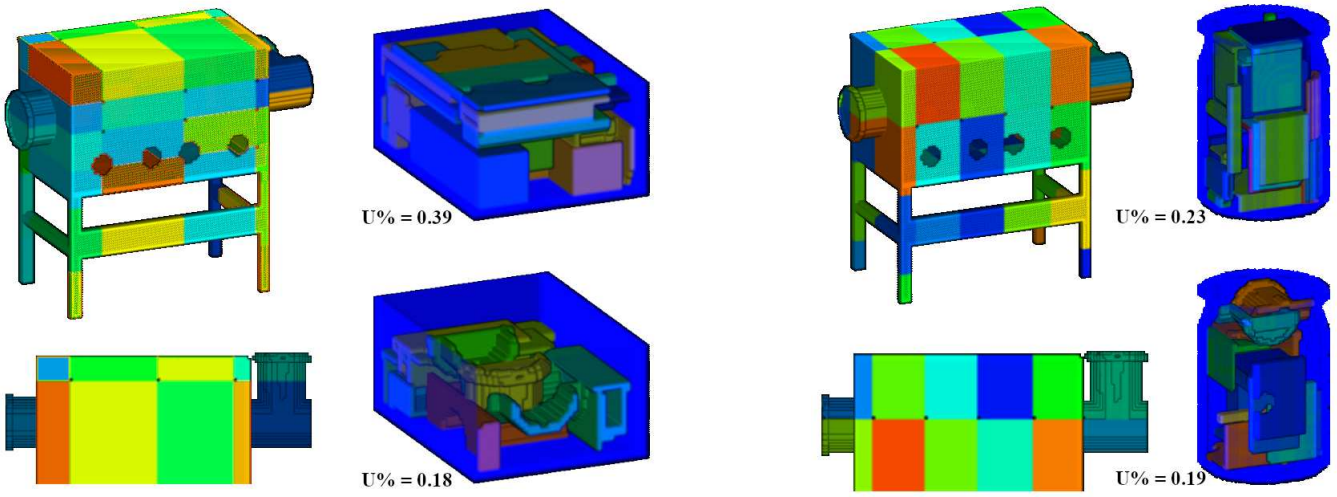
To test the algorithm, we use a glove box structure (Fig. 6a) modified from [13]. We set the dimensions of the structure to 48x112x92 voxels. For packing the shapes, we use a drum container (Fig. 6b top) and a cuboid container (Fig. 6b bottom), which we scale to 43x43x63 and 56x43x27 voxels respectively. For the containers, we set the maximum allowed size for cut parts to $x = 25, y = 25, z = 59$ for the drum container, and $x = 50, y = 38, z = 24$ for the cuboid container.

For cutting, we use zero-width cuts and set the cutting cost weight $w_1 = 1$. For packing cost, we set the cost weight $w_2 = 100,000$ for the drum container and $w_2 = 75,000$ for the cuboid container. This test is analogous to a scenario where a user has selected an appropriate cutting tool for the structure, and must now choose between two types of containers.

V. Results

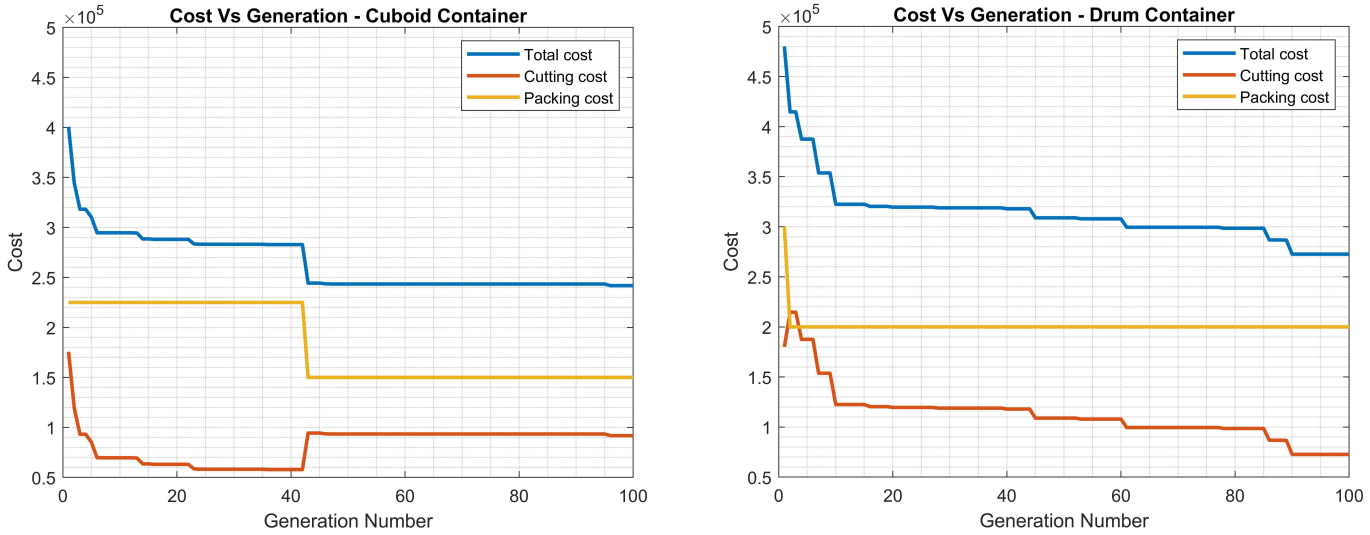
Fig. 7 shows the optimised cutting and packing results for both the drum and cuboid container cases. U% is the percentage space utilisation of each container, calculated by dividing the volume of the packed parts by the available volume inside the container. Fig. 8 shows the cost values (cutting cost, packing cost and total cost) for the best solution in each generation of the GA's run, for both container cases. The run times (rounded to the nearest minute) were 194 minutes for the drum container case and 235 minutes for the cuboid container case.

From Fig. 8 we see that in both cases the total cost decreases throughout the algorithms run, showing that our optimisation methodology is able to minimise total cost. We also see that



(a) (b)

Fig. 7. Optimised cutting and packing solutions. (a) Cuboid container case. (b) Drum container case. U% is the percentage utilisation for each container.



(a) (b)

Fig. 8. Cost values for best solution in each generation. (a) Cuboid container case. (b) Drum container case.

the total cost for the optimised cuboid packing case (2.42×10^5) is lower than for the drum packing case (2.73×10^5). Examining the individual cutting and packing costs, we observe trade-offs made by the algorithm. In both cases, a drop in packing cost corresponds with an increase in cutting cost. This indicates that the algorithm has found a solution involving more cutting to produce smaller parts that can be packed more tightly, reducing the number of containers needed. The savings in packing cost outweigh the increase in cutting cost, leading to a net decrease in total cost.

When comparing individual costs for both cases, the final cutting cost is higher for the cuboid container scenario, while the final packing cost is lower. Since both solutions require two containers, this indicates that the algorithm performed more cutting in the cuboid case to fit the structure into two smaller and less expensive containers. This observation aligns with the higher average packing fraction for the cuboid case (0.29) compared to the drum containers (0.21), as the same structure is packed into containers of smaller volume.

Given that the volume of the structure (30,244 voxels) is smaller than the packing volumes for both the drum and cuboid containers (71,747 and 51,300 voxels respectively), in both cases it is possible to pack the structure into a single container. However doing so would require significantly more cutting, with the increase in cutting cost outweighing the savings in packing cost. To illustrate this, we re-ran the test for the drum case, reducing the cutting weight by a factor of 100 (to 0.01), producing the result in Fig. 9. As observed, the structure can

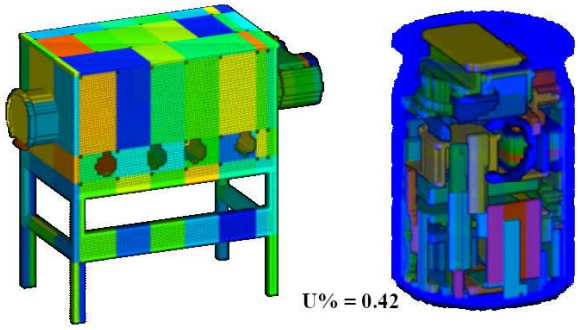


Fig. 9. Optimised solution with low cutting cost weight.

now fit into a single drum container, but only with substantially more cutting. If we multiply the final cutting cost for this solution by 100 (equivalent to performing the same cutting but with a weight of 1 instead of 0.01), we obtain a cutting cost of $6.21\text{x}10^5$ which is approximately 8.5 times higher than the final cutting cost value in Fig. 8b ($7.27\text{x}10^4$). This increase in cutting cost only results in a 2-fold reduction in packing cost. Consequently, the total cost becomes $7.21\text{x}10^5$ which is approximately 2.6 times higher than the total cost in Fig. 8b ($2.73\text{x}10^5$). We note that the percentage utilisation displayed in Fig. 9 (0.42) is the highest possible value achievable with this structure and container combination.

Since some population members are initialized with minimal cutting, the decrease in total cost indicates our optimization approach outperforms a minimal cutting approach. To verify this, we re-ran the algorithm for the drum case with a high cutting cost weight (100 instead of 1). With this high weight, minimal cutting is preferred as cutting cost contributes significantly more to the total cost. For generations 1 and 100, we obtained cutting cost values of $2.34\text{x}10^6$ and $2.30\text{x}10^6$, respectively, with a constant packing cost of $8\text{x}10^5$. Scaling the cutting costs (dividing by 100) to find the equivalent costs with a weight of 1, yields a cutting cost decrease of 400 over 100 generations. Though not as significant as the decrease in Fig. 8b, this still supports our algorithm's effectiveness over minimal cutting.

Notably, the number of containers did not decrease due to the algorithm's focus on cutting cost reduction, which results in large cut parts that don't pack well. While cutting parts smaller would improve packing efficiency (lowering packing cost), it would also increase cutting cost, ultimately outweighing any savings. This suggests that using a cutting optimization algorithm alone to minimize cutting (as in [2], [3]) is inadvisable, as it will likely produce large, poorly packable parts.

VI. Conclusions

With growing interest in seeking optimum strategies for cutting and packing problems in nuclear decommissioning, considering the optimal trade-off between the two processes is a pressing challenge. In this paper, we have presented a novel voxel-based cutting and packing algorithm which addresses this challenge in a feedback fashion using a genetic algorithm to generate cutting solutions that minimize the sum of cutting and packing costs (total cost). By balancing the trade-offs between cutting and packing costs, our algorithm demonstrates its efficacy in minimising total cost. Through our comparative analysis of different decommissioning scenarios, involving two distinct container types, we show that the algorithm can adapt to varying operational conditions and cost structures. Furthermore, our results confirm that the algorithm surpasses the performance of a minimal cutting approach, achieving better overall cost reduction.

This research provides a valuable tool for optimizing the decommissioning process, highlighting its potential to enhance operational efficiency and significantly reduce expenses in nuclear decommissioning projects.

References

[1] X. Jia, Presentation, Topic: "Simulating the Dismantling/Packing Process using NuPlant", Structure Vision Ltd (2009), University of Leeds.

[2] P.-C. Tsai, X. Huang, Y.-H. Hung, C.-H. Huang, and S. Smith, "The computer aided cutting planning of components using genetic algorithms for decommissioning of a nuclear reactor," Annals of Nuclear Energy, vol. 130, pp. 200–207, Aug. 2019, doi: https://doi.org/10.1016/j.anucene.2019.02.041.

[3] J.-Y. Li et al., "The meta-heuristic method based cutting planning for the decommissioning of spallation target in China initiative accelerator driven subcritical system," Annals of Nuclear Energy, vol. 170, p. 108978, Jun. 2022, doi: https://doi.org/10.1016/j.anucene.2022.108978.

[4] Y. Zhao and C. T. Haas, "A 3D Irregular Packing Algorithm Using Point Cloud Data," Jun. 2019, doi: https://doi.org/10.1061/9780784482438.026.

[5] H. C. Kim, S. H. Han, Y. J. Lee, and D. I. Kim, "Packing placement method using hybrid genetic algorithm for segments of waste components in nuclear reactor decommissioning," Nuclear Engineering and Technology, vol. 54, no. 9, pp. 3242–3249, Sep. 2022, doi: https://doi.org/10.1016/j.net.2022.04.004.

[6] M. Yao, Z. Chen, L. Luo, R. Wang, and H. Wang, "Level-set-based partitioning and packing optimization of a printable model," ACM transactions on graphics, vol. 34, no. 6, pp. 1–11, Nov. 2015, doi: https://doi.org/10.1145/2816795.2818064.

[7] J. Vanek et al., "PackMerger: A 3D Print Volume Optimizer," Computer Graphics Forum, vol. 33, no. 6, pp. 322–332, May 2014, doi: https://doi.org/10.1111/cgf.12353.

[8] X. Chen et al., "Dapper," ACM Transactions on Graphics, vol. 34, no. 6, pp. 1–12, Oct. 2015, doi: https://doi.org/10.1145/2816795.2818087.

[9] M. Attene, "Shapes In a Box: Disassembling 3D Objects for Efficient Packing and Fabrication," Computer Graphics Forum, vol. 34, no. 8, pp. 64–76, May 2015, doi: https://doi.org/10.1111/cgf.12608.

[10] A. Martínez-Sykora, R. Álvarez-Valdés, J. A. Bennell, R. Ruíz, and J.M. Tamarit, "Matheuristics for the irregular bin packing problem with free rotations," European Journal of Operational Research, vol. 258, no. 2, pp. 440–455, Apr. 2017, doi: https://doi.org/10.1016/j.ejor.2016.09.043.

[11] K. Fleszar and K. S. Hindi, "New heuristics for one-dimensional bin-packing" Computers & Operations Research, vol. 29, no. 7, pp. 821–839, Jun 2002, doi: https://doi.org/10.1016/s0305-0548(00)00082-4.

[12] Structure Vision Ltd, "NuPlant - New Build and Decommissioning Planning Innovation," www.structurevision.com. [Online]. https://www.structurevision.com/radioactive_waste_disposal.htm (accessed Jun. 29, 2024).

[13] joshgoreworks, "Glove Box Array for Medical or Manufacturing," Thingiverse. [Online]. https://www.thingiverse.com/thing:2072747 (accessed Jun. 29, 2024).