Journal of
Manufacturing and Materials Processing

Article

Exploring Multi-Armed Bandit (MAB) as an AI Tool for Optimising GMA-WAAM Path Planning

Rafael Pereira Ferreira [1,2], Emil Schubert [3] and Américo Scotti [2,4,*]

[1] Federal Institute of Education, Science and Technology of Maranhão (IFMA), São Luis 65030-005, MA, Brazil; rafael.ferreira@ifma.edu.br
[2] Center for Research and Development of Welding Processes (Laprosolda), Federal University of Uberlandia (UFU), Uberlândia 38400-901, MG, Brazil
[3] Alexander Binzel Schweisstechnik GmbH & Co. KG, Kiesacker 7-9, Buseck 35418, Germany; schubert@binzel-abicor.com
[4] Department of Engineering Science, University West (UW), Trollhättan 461 31, Sweden
* Correspondence: americo.scotti@hv.se or ascotti@ufu.br

Abstract: Conventional path-planning strategies for GMA-WAAM may encounter challenges related to geometrical features when printing complex-shaped builds. One alternative to mitigate geometry-related flaws is to use algorithms that optimise trajectory choices—for instance, using heuristics to find the most efficient trajectory. The algorithm can assess several trajectory strategies, such as contour, zigzag, raster, and even space-filling, to search for the best strategy according to the case. However, handling complex geometries by this means poses computational efficiency concerns. This research aimed to explore the potential of machine learning techniques as a solution to increase the computational efficiency of such algorithms. First, reinforcement learning (RL) concepts are introduced and compared with supervised machining learning concepts. The Multi-Armed Bandit (MAB) problem is explained and justified as a choice within the RL techniques. As a case study, a space-filling strategy was chosen to have this machining learning optimisation artifice in its algorithm for GMA-AM printing. Computational and experimental validations were conducted, demonstrating that adding MAB in the algorithm helped to achieve shorter trajectories, using fewer iterations than the original algorithm, potentially reducing printing time. These findings position the RL techniques, particularly MAB, as a promising machining learning solution to address setbacks in the space-filling strategy applied.

Keywords: 3D printing; WAAM; path planning; artificial intelligence; reinforcement learning; multi-armed bandit problem

check for updates

Citation: Ferreira, R.P.; Schubert, E.; Scotti, A. Exploring Multi-Armed Bandit (MAB) as an AI Tool for Optimising GMA-WAAM Path Planning. *J. Manuf. Mater. Process.* **2024**, *8*, 99. https://doi.org/10.3390/jmmp8030099

Academic Editors: Zhichao Liu, Yingbin Hu and Dazhong Wu

Received: 31 March 2024
Revised: 10 May 2024
Accepted: 13 May 2024
Published: 15 May 2024

1. Introduction

Path planning in Gas Metal Arc Welding Wire Arc Additive Manufacturing (GMA-WAAM) has been an ongoing area of research, with numerous strategies developed over time. Each strategy presents a distinct array of advantages and disadvantages. Raster strategies, for instance, boast simple algorithm implementation, yet they suffer from issues like material and heat accumulation, which can be mitigated by introducing idle times between stops and starts [1]. Conversely, zigzag strategies offer advantages in achieving uniform infill layering in relatively simple parts. However, Wang et al. [2] argue that process efficiency declines with this approach, particularly in parts featuring complex geometries prone to arc extinctions, such as internal holes. Similarly, contour strategies exhibit benefits such as effective heat distribution within the part, yet drawbacks highlighted by Ding et al. [1] and Xiong et al. [3] include voids within parts, challenges in very acute angled corners, and centralised heat accumulation leading the parts to residual stress and deformation.

Considering the limitations of these strategies, new approaches have emerged, often building upon previous methodologies. Examples include A-MAT strategies [4], water-pouring strategies [2], and field-based strategies [3]. Additionally, Polygon-division strategies, such as the one introduced by Dwivedi and Kovacevic [5], alongside evolutionary strategies like those developed by Michel et al. [6], have expanded the repertoire of available techniques. This last one has become the product used as the principal path-planning strategy of the WAAM 3D Company, for the time being. A noteworthy advancement is the utilisation of space-filling strategies, commonly employed in polymer 3D printing, which has recently been integrated into the WAAM technology. For instance, the application of the Hilbert strategy (a type of space-filling strategy) in WAAM has been highlighted in the work of Vishwanath and Suryakumar [7]. Moreover, Ferreira et al. [8] showcased another application by employing the Enhanced-Pixel strategy (another approach for using the space-filling strategy) to produce complex WAAM-manufactured parts.

Irrespective of the strategy employed, each one presents its own methodology, challenging the WAAM users to define the approach to choose for their specific part printing needs. Moreover, users face difficulties in selecting the appropriate parameters before initiating the printing process. For instance, in the zigzag strategy, decisions must be made regarding the step-over distances (centre-to-centre bead distances), the angle of the scan lines, and the variation of these angles between layers, among other factors. Similarly, other strategies necessitate decision-making activities, some requiring more decisions than others.

In the realm of WAAM usage, whether by experienced practitioners or newcomers, decision-making regarding printing parameterisation is a constant requirement. The decisions made depend heavily on the user's objectives. At times, expediency may take precedence, prompting the user to prioritise faster part production, even at the expense of quality. Conversely, quality may be paramount for certain projects, guaranteeing a more time-intensive printing process. Similar considerations apply to other aspects of parameterisation. Hence, users must operate with specific objectives in mind, which dictate the selection of strategies or the fine-tuning of parameters within those strategies. Optimisation algorithms offer a promising avenue for streamlining this process. However, the efficacy of such algorithms varies, and the selection of an inappropriate algorithm, such as greedy algorithms, can lead to significant time wastage.

In response to these emerging challenges, integrating artificial intelligence (AI) for dynamic strategies or parameter selection becomes crucial for advancing the field. The primary aim of AI in this context is to identify the optimal options that maximise or minimise outcomes within a specific focus area (the user's objectives). In light of this, machine learning (ML) stands out as a promising field to explore. Considering this, it is feasible to integrate machine learning algorithms into software capable of adaptive execution, employing a self-learning approach to achieve pre-defined objectives. Such integration enables the system to continually adapt and optimise its strategies or parameter selection throughout printing.

As Kumar et al. [9] expound in machine learning, three distinct techniques emerge: supervised, unsupervised, and reinforcement learning. While "supervised" machine learning enjoys widespread adoption, it is tasked with training models to learn functions mapping inputs to outputs using categorised training data. However, it is noteworthy that this method hungers for copious amounts of data to yield precise results, as exemplified in applications like facial recognition employing neural networks. Conversely, "unsupervised" machine learning identifies similarities and extracts insights from unstructured data by clustering shared patterns, which are particularly valuable for organising information, as seen in popular search engine algorithms. However, the preference for "reinforcement learning" (RL) becomes evident when considering its distinctive approach, differentiated from both supervised and unsupervised techniques. It operates without needing pre-existing data-driven relationships. Instead, RL opts for a training approach that rewards desired behaviours while penalising undesirable ones, effectively guiding the learning process. In essence, RL stands out by actively interacting with the environment, a crucial characteristic that sets it apart from supervised learning, which leans heavily on provided sample data or examples. This distinctiveness underscores RL as the optimal choice for scenarios where adaptability and interaction with dynamic conditions are paramount.

Based on the above, this research aimed at exploring if a reinforcement learning algorithm can potentially be a solution to increase the computational efficiency of a WAAM path-planning strategy while keeping the quality performance (to discuss or demonstrate the quality performance concerning potential conformity is out of this work scope, a subject already treated in reference [8], and to be reinforced in coming publications).

2. Background

2.1. Reinforcement Learning Concept

Learning through active engagement with the environment is a core aspect of our comprehension of the learning process. Whether it is children, animals, or newcomers in the workplace, individuals often acquire knowledge by engaging in playful exploration without explicit instructions. Their daily interactions with their surroundings give them valuable insights into causality—the outcomes of their actions, and the strategies required to attain their objectives. Throughout their lives, these interactions with the environment serve as a substantial wellspring of self-awareness and comprehension of the world that envelops them. Reinforcement learning (RL), typically, is a method of learning achieved through dynamic interaction with the environment.

To gain at least a basic understanding of algorithms employing RL, one must acquaint oneself with its key elements and terminologies, as presented in Table 1. At its core, RL centres on the interplay between an agent and its environment. In essence, reinforcement learning problems entail an agent learning through its interactions with an environment to accomplish specific objectives. In contrast to other machine learning approaches, RL introduces a system of rewards (either positive or negative) for every action taken. Positive rewards reinforce good actions, while negative rewards, often framed as regrets, stem from undesirable actions. During each interaction episode between the agent and the environment, the agent executes actions and receives rewards or regrets as a response. As these interactions continue, the agent refines its decision-making process guided by a value function and policy. Furthermore, Ochi and Kamiura [10] assert that an effective policy should strike a harmonious balance between exploration (trying out new actions to discover their effects) and exploitation (leveraging known actions to maximise cumulative rewards) to optimise its overall reward accumulation.

Table 1. Reinforcement learning-related terms and their definitions.

Reinforcement Learning Terms	Definition
Agent	Who or what takes actions to be trained via reinforcement learning
Action	The set of all possible operations/moves the agent can make.
Environment	The place where the agent gathers information, interacts with its surroundings, and acquires knowledge throughout the learning processes.
Reward	A feedback signal provided by the environment to the learning agent (it can be positive rewards, or simply rewards, or negative rewards, or simply regrets).
State	A particular situation in which the environment is at a given moment.

Table 1. *Cont.*

Reinforcement Learning Terms	Definition
Episode	A set of interactions between the agent and the environment that starts from an initial state and ends in a terminal state.
Policy	Set of rules that an agent at a given state must follow to select an action to maximise the reward and avoid regrets.
Value function	The metric used to estimate the expected return or cumulative reward an agent can obtain in a given state.
Exploration	To gather information to understand the environment better.
Exploitation	To use the information from exploration to reach the target results.
Model-based	Type of RL that constructs a representation of the environment to interact with and utilises it to simulate different actions and plan its decisions.
Model-free	Type of RL that does not rely on a pre-defined model, but learns directly from interactions with the environment, only observing rewards and states

This section also presents two practical examples to further clarify the concepts discussed earlier. The first example illustrates model-free reinforcement learning through the training of dogs, while the second showcases model-based reinforcement learning in a maze navigation scenario. In the context of dog training, a reward-based system, which falls under the model-free category, is employed to teach specific commands to the dogs. Through a process of trial and error, the dog learns to associate the commands with particular actions, such as sitting, fetching, etc. Within a household setting (the environment), the training process involves recognising and rewarding the dogs (typically with treats) when they correctly execute the desired actions, while applying penalties or punishments for incorrect behaviours (negative rewards). This repetitive exposure to commands and the reinforcement of rewards and penalties forms a series of training episodes. Over time, this process gradually shapes the dogs' behaviour, leading to the desired responses. It is important to note that not all the concepts and elements discussed earlier are necessarily applied in this example, as their relevance depends on the complexity of the specific training problem at hand.

In the second scenario, there is an autonomous robot acting as the agent, and its task is to navigate through a maze, which serves as the environment. This maze is essentially a grid with walls, open pathways, and a designated goal location. In this case, a model-based approach is utilised, involving the creation of a simulated representation of the environment (the maze). This model captures the inner workings of the maze, allowing the agent to forecast what will happen next and what rewards it can expect based on its current location and the action it intends to take. By having this model in place, the agent can employ planning algorithms to run through various action sequences and assess their outcomes. The model plays a crucial role in helping the agent decide the best actions to take at different points in the maze. In model-based reinforcement learning, the policy often undergoes updates influenced by the results of these planning exercises and the estimated values of other actions at different locations within the maze, a concept known as the value function.

In both scenarios (model-free and model-based), the agent faces the crucial task of striking a balance between exploration and exploitation to achieve its objectives. Exploration encompasses the agent's exercise to execute various actions aimed at acquiring task-relevant information. For instance, this may involve the dog attempting different behaviours to attain rewards or the robot enhancing its comprehension of the maze layout. Simultaneously, exploitation comes into play as the agent uses its learned strategies to take actions that maximise its expected overall reward. By skilfully managing this trade-off between exploring the unknown and leveraging what it already knows, the agent can effectively shape its behaviour (in the dog's example) or successfully navigate the maze (in the robot's example), ultimately reaching the desired goal.

Considering the impracticality of creating models for all potential environments in a space-filling path-planning strategy, mainly due to the varying shapes of the printable parts, it becomes clear that opting for a model-free approach is more appropriate. Among the array of reinforcement learning (RL) methods available, the Multi-Armed Bandit (MAB) algorithm stands out as a promising model-free technique for the application purpose. The MAB algorithm has garnered substantial attention in various real-world applications, spanning recommender systems, information retrieval, healthcare, and finance. Its widespread adoption is primarily attributed to the algorithm's suitability for addressing sequential decision-making challenges involving the recommendation of actions from a set where the reward distribution for each action is unknown. This consideration elucidates why the MAB algorithm was chosen for this work.

2.2. The Multi-Armed Bandit (MAB) Problem

The denomination Multi-Armed Bandit is due to the analogy with a set of slot machines (a slang phrase for slot machine is "one-armed bandit"). Therefore, the technique name is not related to a robbery or a thief, but to the probabilistic chances of winning more times in less time using not the whole set, but selected slot machines over and over. In the literature and various sources from search engines, the MAB problem is often symbolised by an octopus, with each tentacle playing simultaneously with one of a series of side-by-side machines (Figure 1). According to the understanding of the researchers involved in this manuscript, it is symbolic because the concept is a form of reinforcement learning and its derivative techniques, such as the MAB problem, are based on probability. Additionally, having different slot machines with a variable awarding rate is unthinkable, unless they are biased by cheating. Despite this, the symbolism is presented here in Figure 1. First, the octopus uses several slot machines at the same time. However, in sequence, the agent chooses tentacles to continue playing only with the slot machines that were given by chance more rewards (representing lucky machines for a player). It is worth saying that the lucky machines for a player change if the agent starts playing another round (starting other episodes), given the probabilistic character to action (changing the environmental state).

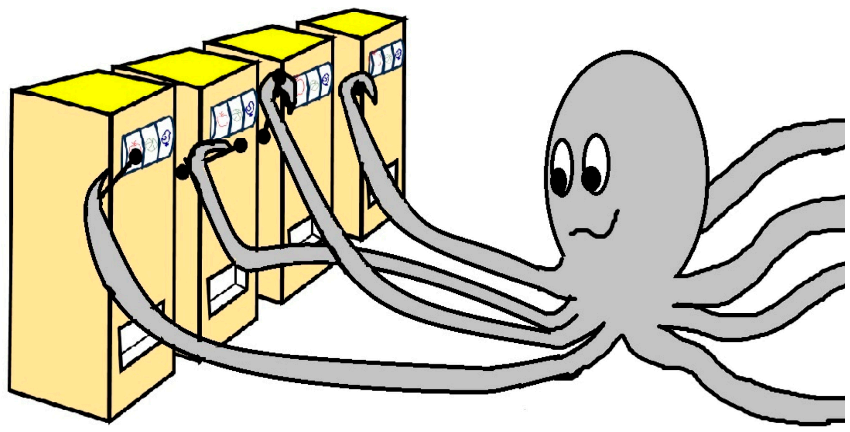

Figure 1. A classical symbolic representation of the Multi-Armed Bandit problem.

A conceptual example of the MAB problem application is the case of online advertising. Imagine a company that wants to determine the most effective online (internet) advertising for a given product. Distinct advertisements (actions) are proposed for the product to be shown to the public using the same media (environment). However, the company has a fixed and limited budget. Click-through rates (famous market metrics nowadays) are a means of quantifying the target public engagement (rewards or regrets). As known, clicks are a marketing metric that counts the number of times viewers have clicked on a digital advertisement. The goal is to allocate the available resources to maximise the clicks (the rewards) from viewers; i.e., to identify and invest more in the most attractive ad for the product. Advertisements with low performance (regrets) will not continue to be displayed. At the start, the company's recommender system (the agent) has limited prior information about the effectiveness of each advertisement. Then, the recommender system iteratively selects advertisements based on a strategy (policy) that balances exploration and exploitation. Initially (first stage), the recommender system may allocate resources to each advertisement uniformly or randomly (exploration). At this stage, the same probability for each ad to be displayed allows it to gather feedback and collect data on the performance of each advertisement. As the recommender system receives rewards (clicks from viewers), it updates (through new episodes) its estimate of the expected compensation or value associated with each advertisement (usually known as value function). Based on the estimated values, the recommender system can recommend the advertisement with the highest value function (exploitation) to display to the viewers.

However, exploration mechanisms must be applied to ensure that most rewarded advertisements (and not the top-ranked only) are given a chance to confirm either their effectiveness or their seasonal character. There are also IA tools for that. According to Almasri et al. [11], the most popular tools used in MAB for solving exploration–exploitation dilemmas are the ε-greedy, the Upper Confidence Bound (UCB), and the Thompson sampling (TS). They help balance exploiting the best-performing advertisement and exploring other options to potentially discover better-performing ones. In dynamic scenarios where the effectiveness of ads can change over time due to shifts in user preferences or market trends, more exploration is required to adapt to these changes and continuously identify the best advertisement to display. By employing the Multi-Armed Bandit approach in this advertising context, the company can optimise its resource allocation and progressively learn which advertisement yields the highest rewards. This allows the company to improve the effectiveness of its advertising campaign through an increase in viewer engagement.

3. Case Study

The trajectory-planning Enhanced-Pixel strategy [8], hereafter referred to as the "original space-filling-based strategy", was taken as a case study due to the complexities involved in ranking combinations of axis ordering and heuristic trajectory in its algorithm reasoning. The multi-combinations of axis ordering and heuristic trajectory are, from now on, treated as "options", for simplicity. The objective of the case was to exemplify the potential of the Multi-Armed Bandit (MAB) problem, a reinforcement learning approach, to optimise computational efficiency in this trajectory planning. It is important to note that alternative path-planning strategies, such as zigzag, contour, etc., could also be considered a case study.

In essence, and limiting to this purpose, the original space-filling-based strategy initiates with the user inputting the desired number of iterations, as depicted in Figure 2. Subsequently, the algorithm iterates through loops, wherein each iteration assesses 10 options (a greedy heuristic routine). Each option generates a trajectory, resulting in a total of 10 trajectories per iteration, which are, then, saved in a matrix. Following this, the algorithm ranks the shortest trajectory distance per iteration from this matrix and stores the outcome in another matrix, denoted as the "Best Value Matrix". This matrix serves as a repository for all the optimal values discovered in each iteration. At this juncture, the algorithm either restarts for a new iteration, repeats the process, or proceeds towards the optimised target value (a detailed description of this Enhanced-Pixel strategy for trajectory planning, including the principle of pixel approach, can be seen in [8]).

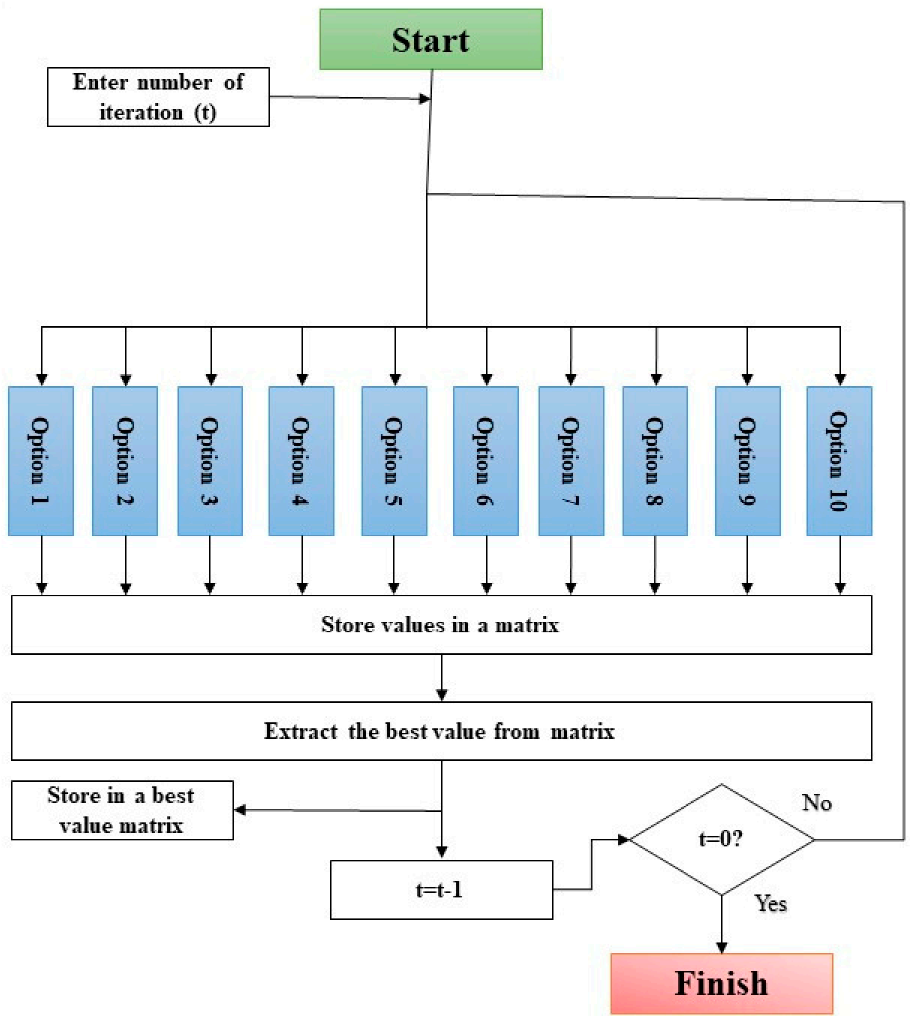

Figure 2. Simplified algorithm flowchart of the original space-filling-based strategy.

Although the algorithm flow work in Figure 2 may potentially identify the best option after all iterations, it requires multiple iterations to converge. Irrespective of the strategy or options, the computational time needed for evaluating each of the 10 options per iteration remains a potential drawback. Consequently, a more intelligent recommendation for optimisation is imperative to augment results and achieve minimised trajectory distances with less computer processing time.

3.1. The MAB-Problem-Based Algorithm

Another algorithm (the environment) configuration, hereafter referred to as MAB-problem-based strategy, was designed to explore the use of a machine learning technique. The intention is to eliminate the need for the conventional greedy routing to rank multiple options per iteration (time-consuming), as the original space-filling-based algorithm version does. This conceived algorithm instead focused on selecting the best of 10 options at each iteration, according to a MAB decision framework (Figure 3). The option recommendation is guided by a policy defining the criteria for selecting the most promising option. As a result, the trajectory distance of only a single option is quantified per iteration, saving computational processing time. Once a trajectory per iteration is generated, it is stored. At this point, the algorithm either restarts for a new iteration (repeating the process) or reaches the end (completing the episode).

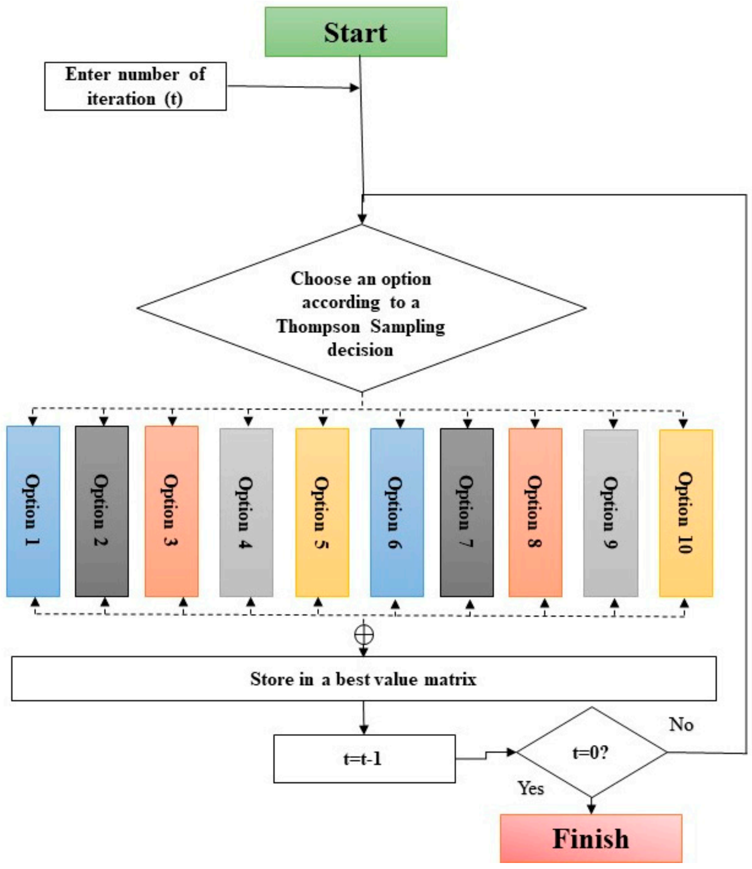
Start
Enter number of iteration (t)
Choose an option according to a Thompson Sampling decision
Option 1
Option 2
Option 3
Option 4
Option 5
Option 6
Option 7
Option 8
Option 9
Option 10
Store in a best value matrix
t=t-1
t=0?
No
Yes
Finish

Figure 3. Simplified algorithm flowchart of the MAB-problem-based strategy.

Thompson sampling was arbitrarily selected as the policy tool for this case study using the Multi-Armed Bandit (MAB) approach, owing to its adept balance between exploration and exploitation, as well as its parameter-free nature. Thompson sampling is particularly useful in scenarios where action rewards follow a Bernoulli distribution, distinguishing successes from failures [12]. This policy is specifically tailored to tackle online decision problems, including the Multi-Armed Bandit problem. It prioritises immediate performance by gathering the latest information to potentially improve future outcomes. The approach involves maintaining a distribution of heuristic performance, selecting the most likely optimal action based on this distribution, and updating it with rewards obtained at each step.

Regarding the application of the Thompson sampling policy tool within the MAB-problem-based algorithm, the probability of combinations evolves as more evidence emerges, resembling a Bayesian approach. In this context, the probability distribution beta distribution was implemented in the policy algorithm. The beta distribution is commonly employed to model proportions or percentages within the interval of $0 \leq X \leq 1$, representing the outcomes of rewards relative to regrets. For further insights into the beta distribution, readers may refer to [13]. However, in brief, each of the 10 options possesses its own probability distribution for being chosen. Upon initialisation of the algorithm, options generating shorter trajectories experience an increase in their alpha parameter of the beta distribution, making them more likely to be selected in subsequent iterations. Conversely, options resulting in longer trajectories see an increase in their beta parameter of the beta distribution, resulting in a decreased probability of selection. This learning mechanism guides the algorithm to exploit options that yield shorter trajectories while also enabling exploration of alternative options.

3.2. Computational Validation

This above approach (Section 3.1) was embedded in the algorithm of the original space-filling-based strategy to assess the potential computational advantages of a MAB-problem-based algorithm, assisted by the Thompson sampling policy, for trajectory distance optimisation. The performance of the MAB-problem-based algorithm was compared with the original algorithm, using both algorithms to plan a WAAM printing of a complex shape (illustrated in Figure 4). An extensive episode of 500 iterations was established for the MAB-problem-based algorithm, to ensure minimally consistent convergencies. It is worth noting that the number of iterations involving the options differs between the two strategies. Specifically, the algorithm using the original space-filling-based strategy evaluates 10 sets of options per iteration to collect 1 best trajectory distance. In contrast, the MAB-problem-based algorithm assesses only one option per iteration to do the same. For the sake of comparison, 50 iterations of the original space-filling-based strategy (50 $\times$ 10 loops) is equivalent to 500 iterations in the MAB-problem-based strategy. Otherwise, if 500 iterations were applied in both approaches, the probability of the optimisation would favour the original space-filling-based algorithm.

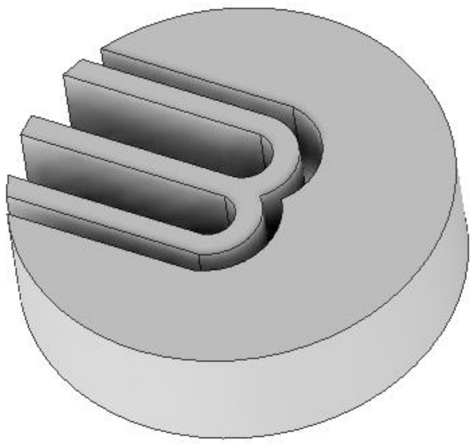

Figure 4. 3D model of a complex shape (a logo) used to validate the use of the MAB problem approach in the trajectory-planning algorithm (nominal dimension of 160 mm diameter and 22 mm high).

Two criteria were proposed and used for the comparison analysis of the computational validation, namely, Convergence Analysis of the trajectory distance and Analysis of Cumulative Regret. Convergence Analysis is standard in optimisation studies, while Cumulative Regret is commonly used for comparison between MAB algorithms. In practical terms, Convergence Analysis examines the speed at which the minimal trajectory distance is achieved for each strategy over the course of iterations. It helps identify the best strategy, based on minimal trajectory distances. Cumulative Regret, the second criterion, represents the accumulated loss resulting from the chosen policy tool not selecting the optimal option. Equation 1 is the way to calculate the progress of this metric along the number of iterations. The ideal option, yielding the best reward up to a given iteration (represented by D^* in the equation) is compared to the actual option chosen at that iteration (represented by $D_{t,h}$). More regrets for the same number of iterations mean fewer positive rewards. Cumulative Regret tells how much one loses by betting more on the wrong option; exploring more than exploiting. The analysis of Cumulative Regret curves follows a similar approach to Convergence Analysis for the minimisation process, with the distinction that regret increases as the number of iterations grows, and higher values indicate larger regrets (poor results).

$$Regret = \sum_{t=1}^{n}[-(D^* - D_{t,h})] \tag{1}$$

The first analysed criterion was carried out through the convergence curves presented in Figure 5a. The comparison between the use of the MAB-problem-based algorithm (dashed orange line) and the original space-filling-based strategy (solid blue line), showed the predominance of the MAB-problem-based algorithm. The MAB-problem-based algorithm converges to a trajectory distance of 2500 mm around the 150th iteration, whereas the space-filling-based strategy converges to a trajectory distance of 2507 mm at around the 280th iteration. This suggest that integrating the MAB algorithm into the process results in shorter trajectories, achieved in fewer iterations compared to the original algorithm. This has the potential to significantly reduce printing time. The curves of the Cumulative Regret criterion are in Figure 5b. It is evident from the part studied that the original space-filling-based strategy (solid blue line) has a higher cumulative regret than the MAB-problem-based algorithm (dashed orange line). This is because the original space-filling-based strategy follows a non-intelligent structure that does not prioritise the best option. Consequently, it can always yield poor results and does not learn from them. This leads to an increase in regret and makes this strategy inefficient. On the other hand, the MAB-problem-based algorithm acquires knowledge about the options that generate the best results as the number of iterations progresses. After evaluating the two criteria, it was found that the MAB-problem-based algorithm performed better than the original space-filling-based strategy.

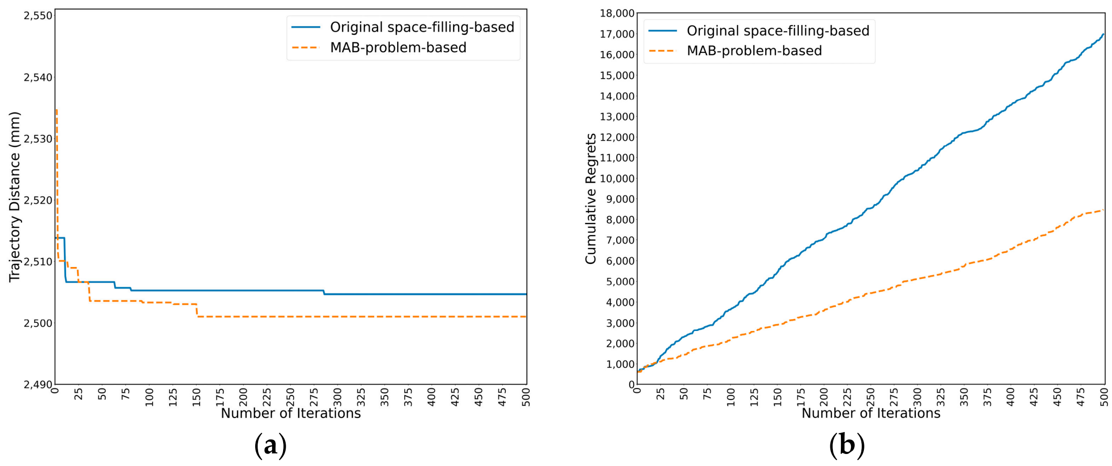

Figure 5. Convergence (**a**) and Cumulative Regret (**b**) analysis charts of the trajectory plans for the studied part (Figure 4).

3.3. Experimental Validation

As suggested in the previous section, the MAB-problem-based algorithm offers notable computational advantages by achieving a short trajectory in fewer iterations than the other strategy (original space-filling-based strategy). However, experimental validation is required to corroborate this computational confirmation. That is, the resultant shorter trajectory reached with a briefer computational time (printing efficiency) must also have printing effectiveness (assessed, for instance, by examining the presence of superficial discontinuities in the printed parts, before and after machining).

Hence, the shape presented in Figure 4 was printed following the experimental settings outlined in Table 2. Due to the shape complexity, the algorithm was adapted to provide a hybrid trajectory strategy plan; i.e., using the contour + MAB-problem-based algorithm. Ten layers were used to build a 22 mm high part (10 layers). The thin part shape contour (forming a mirror image of a long arm B letter) and the body contour (creating a massive C letter) were printed. Considering the printing procedure for practical information, the algorithm generated two sets of continuous contour trajectories, as shown in Figure 6. One contour trajectory for the thin-walled part (resembling the letter B) and one for the bulky body region (resembling the letter C). The first and second trajectory sets, for the odd and even layers, had different start and finish points. The detrimental consequences of arc start–stops, movement accelerations and decelerations, etc., are potentially remedied by the start–stop trajectory changes between layers (odds and evens). These trajectories were, then, replicated for all layers in the printing.

Table 2. The experimental setting for printing the part outlined in Figure 4.

Process	**Kinects** *
Arc welding equipment	iRob 501 Pro (Abicor Binzel)
Torch movement system	ABB Robot IRB 1520 ID
Substrate	SAE 1020 carbon steel (200 × 200 × 12 mm)
Substrate cooling	Natural air cooling—
Wire	AWS ER70S-6—ϕ 1.2 mm
Shielding gas	Ar + 2% CO_2—15 L/min
CTWD **	12 mm
Deposition speed (travel speed)	48.0 cm/min
Set wire feed speed	3.7 m/min
Set Voltage	15.2 V
Set Current	136 A
Printing Z increment	2.2 mm

(*) a Cold Metal Transfer technology from Fronius International GmbH, Wels, Austria); (**) contact tip-to-work distance.

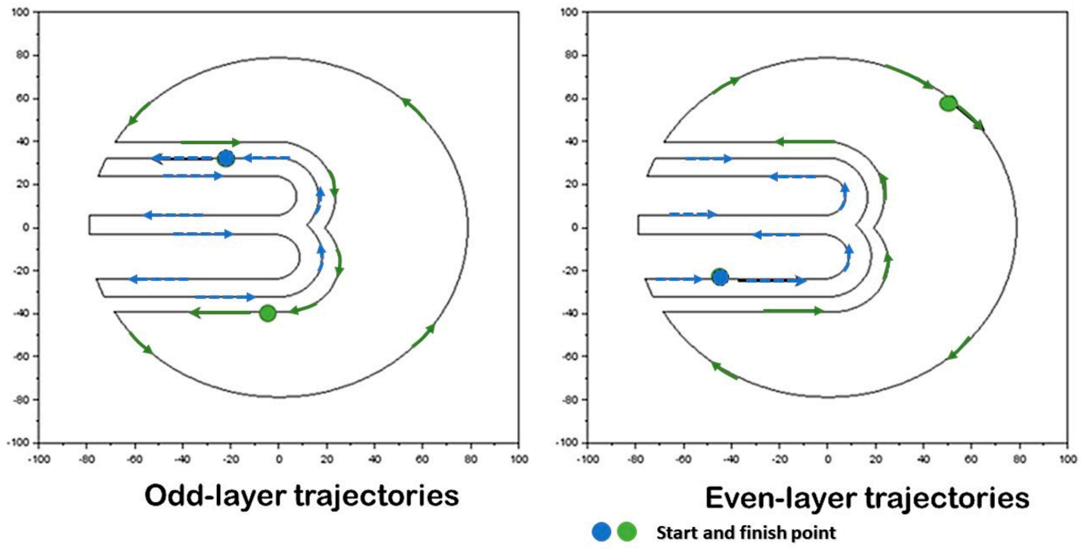

Figure 6. Contour trajectories for the thin-walled and bulky regions.

Figure 7, in turn, displays the trajectory generated to fill in the layers with each bulky body region (resembling the letter C), using the MAB-problem-based algorithm. Two path patterns were created, one for odd layers and another for even layers. Intrinsic to the pixel strategy, different starting points were randomly selected for each odd and even layer. Uniformly distributed surfaces lead to no material accumulation when applying a space-filling approach. On the right frame of Figure 7, the printed model is illustrated. No visible superficial discontinuities, such as voids or lack of fusion, are seen on the rough surface.

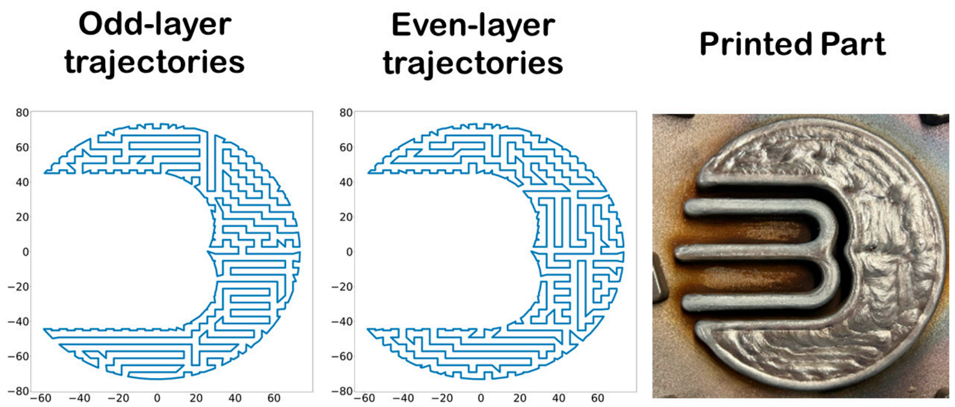

Figure 7. (**Left** and **centre**) trajectory patterns designed by the planner using MAB-problem-based algorithm for odd and even layers; (**right**) corresponding printed part (trajectory starting points are not pointed out because they were randomly selected at each printed layer).

Searching for discontinuities inside the parts, the upper surface of the printed part used in this case study was flattened by machining (removing approximately 5 mm of material), as illustrated by Figure 8. The part printed after the trajectory planned with the MAB-based pixel strategy did not provide any visible internal discontinuity on the machined plane.

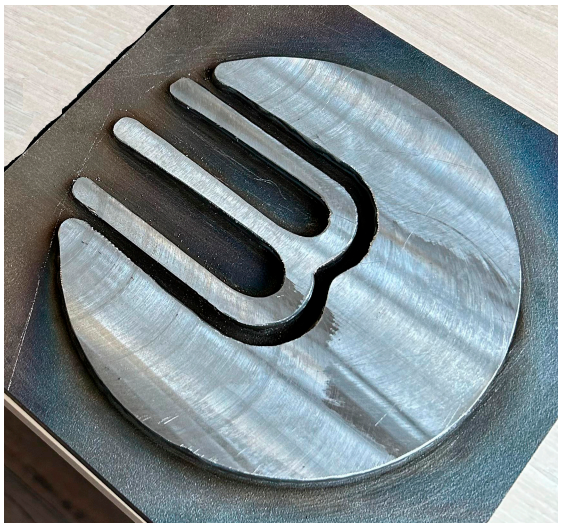

Figure 8. The printed part with the upper surface machined.

4. Conclusions and Future Work

The objective of this investigation was to contribute to the path-planning developments for WAAM. The target was to understand the concept and explore whether a reinforcement learning technique, namely the Multi-Armed Bandit (MAB) problem, can potentially be an artificially intelligent (AI) solution to increase the computational efficiency of a WAAM path-planning strategy. Applying the MAB problem concept in the algorithm for the trajectory planning of a space-filling-based strategy (Enhanced-Pixel), assisted by the Thompson sampling policy tool, both printing efficiency (quantified by the Convergence Analysis of the trajectory distance and analysis of Cumulative Regret) and printing effectiveness (assessed by examining the presence of superficial discontinuities in printed complex parts), were evaluated. The results showed that this approach is efficient (reduced computational processing time due to fewer optimisation iterations than the original algorithm) and effective (it provided a short trajectory that supported printing a part without any superficial discontinuity).

As previously discussed, this approach can be extended to encompass other trajectory-planning strategies, incorporating optimisation objectives beyond simply finding the shortest trajectory or printing time. Future research will explore integrating factors such as the number of starts and stops, acceleration/deceleration profiles, heat accumulation, and residual stresses into the optimisation framework. Therefore, the future purpose is to develop and validate more comprehensive Multi-Armed Bandit (MAB)-based algorithms.

Before closing, it is important to make the reach of this work more transparent. Printing time does not depend on the approach to using the Multi-Armed Bandit (MAB) tool. This approach claims to increase the computational efficiency of a WAAM path-planning strategy to generate optimal trajectory while maintaining quality performance, but not to increase the printed part quality. This paper introduced this tool to WAAM trajectory planning, confirming its performance. Printing time and part quality, in turn, depend on the trajectory and parameters. Optimising the trajectory and parametrisation are, in fact, the means of reducing printing time and reaching printed part quality, respectively. In this work, the trajectory was optimised by using the Enhanced Pixel strategy, presented elsewhere [8], through hyper-heuristic loops. Therefore, regardless of the strategy, conventional (zigzag, contour, raster, etc.) or space-filling (such as pixel) MAB can sort the most efficient trajectory with a reduced number of iterations, but not modify the trajectory efficacy.

Author Contributions: Conceptualisation: R.P.F. and A.S.; methodology: R.P.F. and A.S.; software: R.P.F.; validation: R.P.F. and A.S.; formal analysis: R.P.F., E.S. and A.S.; investigation: R.P.F.; resources: E.S.; data curation: E.S. and A.S.; writing—original draft preparation: R.P.F. and A.S.; writing—review and editing: R.P.F., A.S. and E.S.; visualisation: R.P.F. and A.S.; supervision: R.P.F., E.S. and A.S.; project administration: E.S. and A.S.; funding acquisition: E.S. and A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Council for Scientific and Technological Development—CNPq (306053/2022-5) and the Coordination for the Improvement of Higher Education Personnel—CAPES (88887.696939/2022-00).

Data Availability Statement: The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Acknowledgments: The authors would like to thank the Federal University of Uberlandia, Brazil, and Alexander Binzel Schweisstechnik GmbH & Co. for their generous support in providing laboratory infrastructure and essential materials, which significantly contributed to the success of this research.

Conflicts of Interest: Author Emil Schubert is employed by Alexander Binzel Schweisstechnik GmbH & Co. The remaining authors, from educational institutions, declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Ding, D.; Pan, Z.; Cuiuri, D.; Li, H. A practical path planning methodology for wire and arc additive manufacturing of thin-walled structures. *Robot. Comput.-Integr. Manuf.* **2015**, *34*, 8–19. [CrossRef]
2. Wang, X.; Wang, A.; Li, Y. A sequential path-planning methodology for wire and arc additive manufacturing based on a water-pouring rule. *Int. J. Adv. Manuf. Technol.* **2019**, *103*, 3813–3830. [CrossRef]
3. Xiong, Y.; Park, S.; Padmanathan, S.; Dharmawan, A.G.; Foong, S.; Rosen, D.W.; Soh, G.S. Process planning for adaptive contour parallel toolpath in additive manufacturing with variable bead width. *Int. J. Adv. Manuf. Technol.* **2019**, *105*, 4159–4170. [CrossRef]
4. Ding, D.; Pan, Z.; Cuiuri, D.; Li, H.; Larkin, N. Adaptive path planning for wire-feed additive manufacturing using medial axis transformation. *J. Clean. Prod.* **2016**, *133*, 942–952. [CrossRef]
5. Dwivedi, R.; Kovacevic, R. Automated torch path planning using polygon subdivision for solid freeform fabrication based on welding. *J. Manuf. Syst.* **2004**, *23*, 278–291. [CrossRef]
6. Michel, F.; Lockett, H.; Ding, J.; Martina, F.; Marinelli, G.; Williams, S. A modular path planning solution for Wire + Arc Additive Manufacturing. *Robot. Comput. -Integr. Manuf.* **2019**, *60*, 1–11. [CrossRef]
7. Vishwanath, N.; Suryakumar, S. Use of fractal curves for reducing spatial thermal gradients and distortion control. *J. Manuf. Process.* **2022**, *81*, 594–604. [CrossRef]
8. Ferreira, R.P.; Vilarinho, L.O.; Scotti, A. Enhanced-pixel strategy for wire arc additive manufacturing trajectory planning: Operational efficiency and effectiveness analyses. *Rapid Prototyp. J.* **2024**, *30*, 1–15. [CrossRef]

9. Kumar, S.; Gaur, V.; Wu, C. Machine learning for intelligent welding and manufacturing systems: Research progress and perspective review. *Int. J. Adv. Manuf. Technol.* **2022**, *123*, 3737–3765. [CrossRef]
10. Ochi, K.; Kamiura, M. Overtaking method based on sand-sifter mechanism: Why do optimistic value functions find optimal solutions in multi-armed bandit problems? *Biosystems* **2015**, *135*, 55–65. [CrossRef] [PubMed]
11. Almasri, M.; Mansour, A.; Moy, C.; Assoum, A.; Le Jeune, D.; Osswald, C. Distributed competitive decision making using multi-armed bandit algorithms. *Wirel. Pers. Commun.* **2021**, *118*, 1165–1188. [CrossRef]
12. Jain, S.; Bhat, S.; Ghalme, G.; Padmanabhan, D.; Narahari, Y. Mechanisms with learning for stochastic multi-armed bandit problems. *Indian J. Pure Appl. Math.* **2016**, *47*, 229–272. [CrossRef]
13. Gupta, A.K.; Nadarajah, S. *Handbook of Beta Distribution and Its Applications (Statistics: A Series of Textbooks and Monographs)*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2004; ISBN 978-0824753962.