Industrial project: Esautomotion

First report

Gialcuca D'Addese, Giorgia Franchini, Matteo Magnani, Carmelo Scribano

PART I

Introduction

The process of bending stands as one of the most crucial steps in the realm of sheet metal prototyping. It encompasses the strategic formulation of bending sequences and the judicious selection of bending tools, both of which are pivotal stages within this manufacturing procedure. Historically, these two undertakings relied on manual procedures. In recent years, unfortunately, the experience of operators is being lost due to the continuous process automation. However, the rapid advancements in computer hardware and software in recent years have ushered in the era of computer-aided methods for executing these tasks [4, 6]. Nonetheless, when compared with a substantial number of bending operations, the sheer multitude of possible combinations for bending sequences presents computational challenges. Furthermore, when the planning of sequences involves multiple bending steps, the complexity of the situation inevitably escalates. The number of possible combinations of fold sequences is $n!$, so using brute force algorithms to approach the problem can work for pieces that have 3 or 4 total folds [18]. However, when the number of folds goes above 10, the number of combinations exceeds three and a half million. By implementing a set of rules used by industry professionals, this research space can be constrained, thereby limiting the sequence selection possibilities [2]. However, these rules alone are often insufficient to achieve an acceptable computational time.

I
SEQUENCE SELECTION

The problem can be detailed as follows. The objective is to obtain any type of part i(b) that can be achieved by folding at specific points on an initially flat piece of metal i(a). To obtain this part, it is necessary to apply the folds one by one at predetermined points as shown in the diagram [flat piece with folding points]. The challenge of this task arises when a significant portion of folding sequences becomes infeasible due to collisions with equipment parts, as depicted in the figure i(c). Therefore, it is imperative to employ an algorithm capable of finding the correct folding sequence that avoids collisions with the

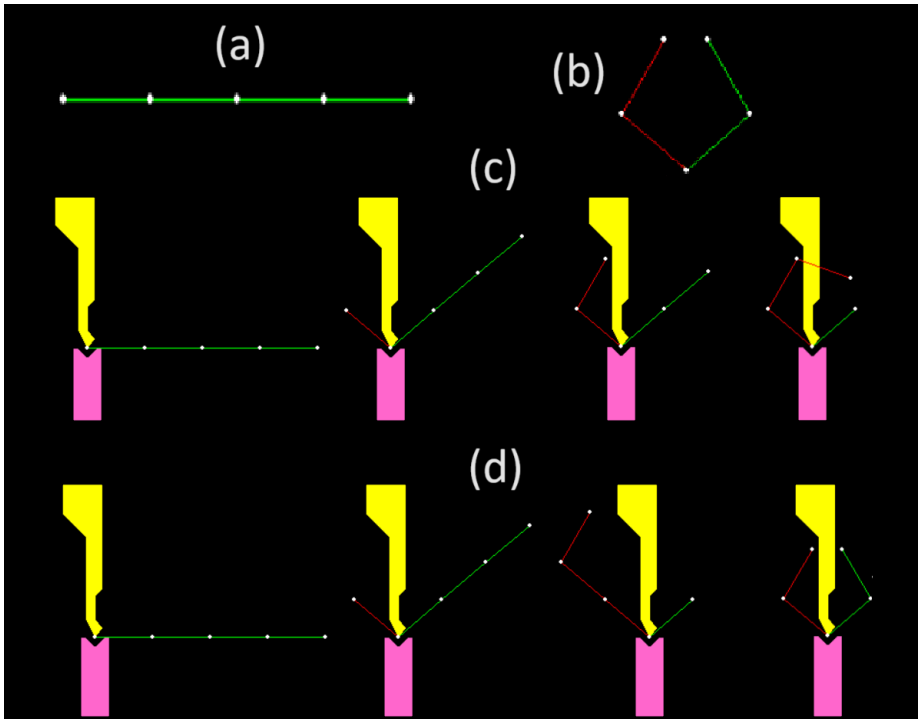


FIG. i In the figure, we have the component before the bending process (a) and after the complete bending (b). Subsequently, there are two bending sequences, one unfeasible (c) and one correct (d).

tools i(d). Given the need for the operator to reposition the piece or change tools between folds, it is also important to identify among collision-free solutions the one that minimizes the time required (or the one with the fewest tool changes or rotations of the metal piece).

II
TOOL SELECTION

The V-bending procedure is executed on the press brake, involving the utilization of its mobile and fixed components, where the punch descends upon the sheet metal onto the die with appropriate force. Various categories of V-bending tools, including sash, gooseneck, and narrow relief punches, as well as distinct V-dies, are available. Consequently, the choice of a punch and die for conducting V-bending along a specific bend line necessitates adherence to the technological constraints outlined in [19]. Following the satisfaction of these technological constraints, it is imperative that these tools facilitate a collision-free bending process.

Owing to the interaction between the bending sequence and the tool selection in the V-bending process, a collision may arise using a particular set of bending tools, while it may not occur with an alternative set. For instance, the detection of collisions takes place in V-bending processes on a bent part, depicted in figure 2(a), utilizing two distinct bending punches shown in figure 2(b). Although a collision occurs during the bending of b2 when employing the first punch, as indicated by the bending sequence b1, b3, b2 in figure 2(c), no collisions arise when employing the second punch with the same sequence, as depicted in figure 2(d).

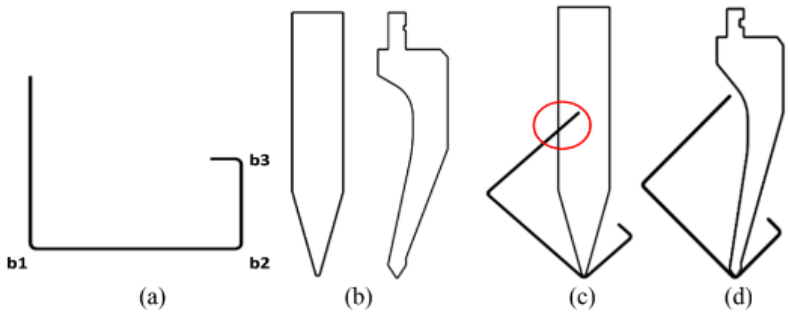


FIG. ii In this image, it is possible to evaluate how for the bending of specific form (a), it is necessary to choose a punch that is suitable for the shapes the component can assume.

III
OPERATIVE PROPOSALS

As the quantity of bends grows, the potential count of feasible bending sequences for a sheet metal component rises as well. Nevertheless, it's important to note that not every single one of these conceivable bending sequences is practicable. Among the potential scenarios, there might be instances of the sheet metal component self-intersecting during the bending process, instances where the sheet metal component interferes with the bending tool, and situations where interference arises between the sheet metal component and the bending machine. If one were to undertake the verification of each conceivable sequence, it would necessitate a substantial investment of computation time. The objective is to seek an algorithm capable of solving the presented problem within acceptable timeframes.

The project will be divided into increasingly challenging problems:

- Finding a feasible bending sequence for a 2D part given a punch and a die.

- Finding a feasible bending sequence for a 2D part while having access to a set of punches and dies and maintaining them throughout the entire bending process.

- Finding a feasible bending sequence for a 2D part while being able to alternate between different punches and dies during bending.

- Finding an optimal bending sequence for a 2D part.

Once we have successfully completed the project with 2D pieces, one of the next milestones to achieve will be the resolution of 3D pieces. The difficulty of the problem will grow exponentially as we will need to consider all possible rotations of the piece in three dimensions. So, in a second phase of the project, we will expand the program to address the following points:

- Finding a feasible bending sequence for a 3D part.

- Finding an optimal bending sequence for a 3D part.

Therefore, we have elected to conduct a literature review to explore the most prevalent methods utilized for addressing this type of problem. Numerous methodologies have been employed for solving this problem. The first approach found to the problem are genetic algorithms (GAs) [22] occasionally coupled with neural networks [25].This type of algorithm, through the natural selection of the most suitable individuals, leads to optimal solutions over multiple generations, starting from random individuals. A different approach is to attempt to reduce the number of folds by combining consecutive folds into a single one. This is done when a sequence of few folds forms shapes known as the L-shaped fold or the U-shaped fold or the S-shaped fold [12]. To conclude, one of the most efficient approaches to the problem is that of heuristic algorithms. This involves the creation of a mathematical model that can reflect the problem's constraints, as well as the use of branch and bound algorithms where an exhaustive search of a tree with all possible combinations is performed using specialized search methods tailored to the problem [5, 11, 13, 17].

Consequently, we have chosen to tackle the problem using two algorithms previously employed, namely, genetic algorithms and branch and bound. Additionally, we have introduced a novel third method, reinforcement learning, which we have aimed to put to the test for this particular problem type.

In the following chapters, we will address the pros and cons of these algorithmic types, delving into the current state of the art regarding these approaches in relation to the presented problem. Subsequently, we will present our evaluations concerning the methodology.

PART II

Genetic Algorithm approaches

I
STATE OF THE ART

Genetic Algorithms were initially developed both as an emulation of "physical adaptation" and as a solution for practical optimization problems do not require differentiation of the objective function representing cost or behavior. GAs are grounded in the creation of a population of individuals (chromosomes) to which essential operations including selection, crossover, mutation, and recombination are applied.

GAs function in accordance with Darwin's principle of natural selection, famously known as 'survival of the fittest'. GAs approach problem-solving not through a mathematical lens, but rather through a 'biological' perspective. They possess the capacity to determine the optimal or near-optimal solution regardless of the nature of the problem, which might encompass non-linearity, discrete-time considerations, multiple optimal points, conformity to equal or unequal constraints, and even non-NP completeness [7].

GAs assume that each conceivable solution can be represented (encoded) as a set of parameters, akin to genes within a chromosome, often structured as sequences of binary digits. The 'fitness value' gauges the 'quality' of a solution proposed by the chromosome for the specific problem. Through genetic evolution, chromosomes with higher fitness values tend to generate offspring of superior quality, i.e., better solutions for the problem.

In a practical application of GAs, a population of chromosomes is formed by randomly selecting potential individuals. The size of this population can vary based on the problem. During each cycle of the genetic process (referred to as the 'evolution process'), the subsequent population (generation) arises from the chromosomes of the current population. This involves selecting certain chromosomes (parents) and subsequently combining and recombining parental chromosomes to produce offspring for the next generation. It is expected that, through this evolutionary process, the 'fittest' chromosome will generate a greater number of offspring, thereby having a better chance of survival in successive generations, effectively mirroring the principle of 'Survival of the fittest' [7].

The most basic form of GA comprises three operators: selection, crossover, and mutation. Selection determines which individuals are used for reproduction, with more 'adaptive' chromosomes having a higher likelihood of repeated selection for reproduction. Crossover randomly selects a position within the chromosome and swaps the sub-sequences (before and after this position) between two individuals to create two offspring Figiii.

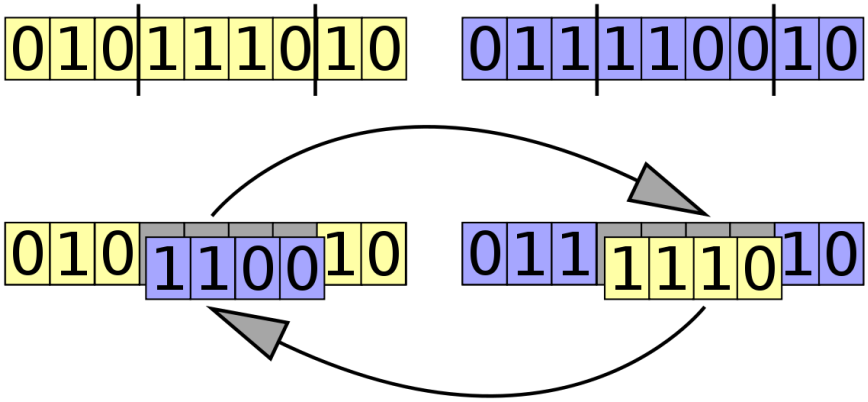
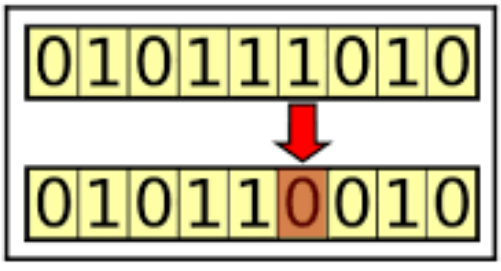


FIG. iii Example of two-point crossover with binary chromosomes.

Mutation introduces random changes to some genes of a chromosome, typically after crossover Figiv. Mutation may affect any gene within the chromosome with a usually small probability, e.g.0.1%. The cycle of evolution is iterated until a particular criterion is met. This criterion can encompass the count of evolution cycles (generations), the frequency of individual changes across generations, or a designated value of the objective function.



Mutation operator applied to a binary-coded chromosome

FIG. iv Mutation on a gene occurring subsequent to the crossover event.

How can we find in [22] the chromosomes contained by each individual contain information about the sequence of bends, the punches, and the utilized dies. This set of information corresponds to a single individual. Initially, each individual starts with a random series of bends and punches, as well as dies.

The fitness value of the chromosomes is assessed, and an examination is conducted to determine the presence of an optimal solution. In cases where optimal solutions are not generated, these chromosomes are chosen for reproduction. The roulette selection method is employed to designate chromosomes for reproduction. During selection, the fitness values are treated as probabilities, with chromosomes possessing higher fitness values being accorded a greater likelihood of being chosen.

The fitness function takes into consideration various parameters, [9] including the execution time of the piece, the number of times the piece needs to be rotated, the amount of sheet metal held by the operator during bending, and so forth. However, having a bending sequence that results in a collision significantly reduces the fitness value of the individual.

II
WEAKNESS OF THE GENETIC ALGORITHM

In terms of strengths, genetic algorithms offer the advantage of being able to work with a set of individuals without requiring prior knowledge of the bending rules for the pieces. Each individual is assigned a fitness value based on parameters that the operator deems important. This implies that the bending process still needs to be tested to identify potential collisions with the machinery, but there is zero prior knowledge of the bending process. This simplicity on the bending processing end eliminates the need for optimization callbacks for the process. Instead, for each individual, the bending process is tested, and its feasibility is assessed.

On the other hand, genetic algorithms prove to be a rather complex solution for this type of problem. While their strength lies in their ability to function in scenarios where planning the various steps in advance is not possible, in a problem like this where it's feasible to anticipate the issues encountered in upcoming bends, their performance isn't as efficient.

As the number of bends increases, the sheer quantity of randomly generated initial individuals at the outset escalates significantly, making it highly improbable to even have a single collision-free individual. Consequently, attempting to combine two individuals through crossover that are not feasible solutions is unlikely to yield offspring with collision-free bending sequences. At this juncture, the algorithm will strive to optimize the found individuals

by expediting other parameters, such as the time taken. However, this process tends to lead not to collision-free solutions, but rather to the continued optimization of non-feasible solutions. It is conceivable that, through crossover and mutations, the process could eventually arrive at acceptable solutions. Nevertheless, this progression might necessitate several generations, and therefore, a considerable amount of computational time.

Conversely, the scenario changes when the initial pool of individuals is not random. Feasible solutions derived from other computational algorithms could be utilized as the starting population for the genetic algorithm. This approach may potentially enable the optimization of various properties deemed essential for fitness.

III
OPERATIVE PROPOSALS

Considering the project is divided into multiple part, during the initial operational phase where we will be working on 2D parts using a single punch and die, genetic algorithms appear to be a solution type that is overly complex. We have opted to temporarily suspend the implementation of this algorithm for now, planning to resume it at a later stage when addressing issues that require material change during the bending process.

PART III

Branch and Bound approaches

I
STATE OF THE ART

Branch and Bound (B&B) [10], is an algorithmic approach used for solving optimization problems, particularly in combinatorial optimization domains. It explores the solution space by breaking it down into smaller subproblems, making informed decisions to eliminate certain branches based on bounds, and then recursively solving the remaining subproblems. This strategic methodology substantially reduces the search scope, leading to more efficient solutions for difficult problems.
Here is a description of the B&B strategy:

1) **Divide et impera**: B&B divides the initial problem into smaller and easier subproblems, often by segmenting the problem space into subsets or accounting for different constraints and variables.

2) **Exploration of subproblems**: the algorithm progressively addresses the subproblems by solving them recursively. It determines which subproblem to tackle next based on a heuristic or a lower bound estimation, which plays a crucial role in enhancing search efficiency.

3) **Bounds**: bounds are critical for guiding decisions regarding which subproblems to explore and which to prune. They provide estimations of the optimal solution value within a given subproblem:

   - *Upper Bound (UB)*: sets the upper limit for the optimal solution within a subproblem.
   - *Lower Bound (LB)*: establishes the lowest value for the optimal solution within a subproblem, often derived from heuristics or relaxation methods.

4) **Pruning**: pruning involves discarding subproblems that are proved to not contain the optimal solution. If the lower bound of a subproblem is higher than the upper bound of another subproblem, the latter subproblem is pruned, as its solution cannot outperform the known solution.

For recent advance and application of B&B technique, refer to [14].

In the following paragraph we summarize some of the research papers which deal with the problem of bending, using B&B techniques. In general, there are two methods used to construct bend sequencing models in the context of sheet metal bending:

Forward Planning Approach: In the forward planning approach, the sequence of bends is determined by iteratively adding one bend at a time. Starting with the initial flat state of the sheet metal, each subsequent bend is selected based on the existing bent configuration. This method progresses step by step, adding bends sequentially until the desired final shape is achieved. Forward planning can be intuitive as it follows the chronological order of bends, but it might require extensive simulations to validate the feasibility of the sequence.

Backward Planning Approach: In contrast, the backward planning approach involves working from the final desired shape of the part back to the initial flat state. It begins by identifying the necessary sequence of bends that would lead to the desired shape. This approach can potentially reduce the number of possibilities to explore and can be advantageous for identifying infeasible sequences earlier in the process.

We split the state of the art analysis following these two strategies for solving the problem.

Forward Planning

The arrangement of bends is established through a step-by-step process of gradually introducing individual bends. Originating from the initial flat state of the sheet metal, each consecutive bend is chosen by considering the prevailing bent arrangement. This approach advances incrementally by incorporating one bend at a time.

- [8]: This paper employs a distributed planning architecture, the system currently comprises a central operation planner and three distinct domain-specific planners: tooling, grasping, and moving. The central operation planner generates multiple alternative partial sequences, through heuristics and path search on graph, which are then evaluated by specialized planners based on their respective objective functions. State-space search techniques are employed by the central operation planner to optimize the sequence of operations.

- [3]: The proposed algorithm is based on the search for a path that minimizes costs, passing through all the nodes of a graph representing the bends needed to solve the problem. The costs are dynamically calculated at each step.

- [17]: The developed algorithm segments the part into fundamental shapes like channels and spirals, deriving partial sequences corresponding to each. The overall bending sequences for the complete part are constructed by combining these partial sequences. Various strategies are implemented to minimize solutions and, consequently, search time during the combination process. All sequences, both partial and complete, undergo validation for potential part-tool collisions and adherence to tolerance constraints. Subsequently, the sequences are arranged based on the total process time, and a robot is utilized to ensure precision in handling operations, achieving the desired process time accuracy. In the final step, the sequence associated with the shortest process time is identified as the optimal solution.

- [12]: The initial step involves the definition of fundamental bending patterns (6 different, L-shape, U-shape,...), each of which is characterized by a specific set of operation rules. The sheet metal is subsequently divided into a sequence of these bending patterns, which then informs the development of bending sequences. To determine the suitable bending tools, the process combines the contours of each bending operation, selects the appropriate bending punches from the bending-tool database, and carries out an interference check with the bending contours.

- [23]: The heuristic algorithm operates by computing formulas designed to assess geometric feasibility. Its objective function aims to minimize several parameters, including tool changes, tool stroke, changeover, and workpiece placement.

Backward search

Backward planning approach explore feasible "straightening" sequences based on the part's final bent state. This method is smart because it identifies and eliminates impractical sequences "early", especially due to potential interferences near the final state.

- [20]: The algorithm utilizes the $A*$ graph search algorithm [15] to optimize the bending process represented as a tree structure. Additionally, the research introduces specialized cost and heuristic functions specifically designed for the bending problem. These functions take into account various factors such as tool changing, product manipulations (translation, rotation), stability, and the imposition of penalties based on precedence constraints throughout the optimization process.

- [24]: the proposed algorithm for determining viable bending sequences involves simulating bends on the initial geometric model of a part. If constraints are satisfied, bending occurs; otherwise, the search on the tree stops. The process continues until a terminal node is reached, yielding a viable straightening sequence. The algorithm selects optimal punch and die pairs in advance to meet bending requirements. Tool accessibility is verified by checking for interferences between tool models and the sheet metal part. The process also detects self interferences within the part's plates and examines interferences between tool models and component plates during different bending rotation steps.

- [13]: a Pareto optimization with a constraint-based approach is proposed, involving the utilization of predefined constraint types, including precedence, neighborhood, resource assignment, and sharing, in conjunction with geometric constraints.

- [11]: the algorithm operates by evaluating the feasibility of bending process plans through a simulation process. In addition to identifying collisions, the algorithm incorporates a set of hard reject criteria for assessing the suitability of the bend sequence. These criteria include the availability of appropriate gauging edges for precise part positioning, adherence of part dimensions to specified tolerances, and the front extend ratio. The algorithm systematically examines each bend sequence within the simulation. It checks whether the plan satisfies the conditions specified by the rejection criteria. The $A*$ algorithm for path selection is utilized. This forms an evolving front of open nodes, where each node's cost to the root and estimated cost to the target are assessed. The node with the lowest $f(n)$ value, considering bending operation costs, is selected.

- [16]: a two-stage algorithm is designed. In the initial stage, a bend feasibility Matrix is created, mapping the entire search space using a geometric approach. This matrix facilitates swift determinations regarding the manufacturability of the part with the given tools. The subsequent

stage employs a best-first search algorithm operating on a graph to pinpoint the bend sequence. This algorithm eliminates the need for costly collision tests, as these computations are already performed in the first stage. The algorithm's performance is compared with that of a GA. The results demonstrate the superiority of the best-first search algorithm over the GA in addressing the bend sequencing problem.

- [1]: It employs the $A*$ graph search algorithm for optimization on tree. Furthermore, the study introduces a cost function and a heuristic function tailored for the wire-bending problem (considering rotation of the tool, rotation of the wire), along with a method to parallelize the execution of $A*$ (path matrix approach, divide et impera philosophy).

II
CHARACTERISTICS OF B&B

The B&B strategy offers several advantages and drawbacks in the process of optimization problem-solving.

On the positive side, one of the most notable benefits of the B&B approach is its ability to guarantee finding the optimal solution. By systematically exploring the search space and utilizing bounds to eliminate unpromising branches, B&B ensures that the final solution is the best possible outcome. In terms of quality, the algorithm not only provides the optimal solution but also establishes a lower bound on that solution's value. Even if the search is not exhaustive, this bound offers valuable insight into the quality of heuristic solutions or serves as an indicator of progress.

However, B&B does come with its share of challenges. One significant drawback lies in the computational complexity. The more big the tree is, the more algorithm's computational demands can raise, leading to increased processing time and memory usage. Then, generating accurate tight bounds can also be difficult. The success of the approach heavily relies on the quality of heuristics employed to estimate these bounds and guide branching decisions.

In summary, the B&B strategy offers the advantage of guaranteed optimality and adaptability across domains. However, it also demands careful consideration of heuristics, bounding strategies, and problem-specific adjustments. While B&B is effective for solving optimization problems, how well it works relies on the problem's features, the quality of the methods used, and how it's adapted to fit the problem. This approach strikes a balance between finding the best solution and the difficulty of the calculations, making it useful for many kinds of optimization tasks.

III
OPERATIVE PROPOSALS

Due to the inherently combinatorial nature of the sheet bending problem, wherein decision permutations can reach a factorial scale of $n!$, with n representing the number of required bends for each sheet, the utilization of the B&B strategy emerges as an optimal method for addressing this specific problem.

PART IV

Reinforcement learning approaches

I

STATE OF THE ART

Reinforcement Learning (RL) [21] is a subfield of machine learning that aims at defining intelligent agents capable of learning how to make sequential decisions by interacting with an environment in order to maximize an expected reward. Reinforcement Learning algorithms have found a multitude of applications across diverse domains, showcasing their versatility and efficacy in solving complex problems. A vast panorama of different RL techniques have been applied with remarkable results in complex scenarios such robotic, autonomous vehicles, finance, and even healthcare. Of interest for our domain, RL techniques are employed to efficiently navigate complex solution spaces and find optimal configurations, and have been broadly applied in manufacturing scenarios for optimizing production processes for minimal downtime and maximum efficiency. To the best of our knowledge, no prior art exists on the application of RL algorithms to the problem of process planning for metal sheet bending. Therefore, hereafter, we provide a small introduction to the principles of RL and propose a generic framework to model our problem as a reinforcement learning task.

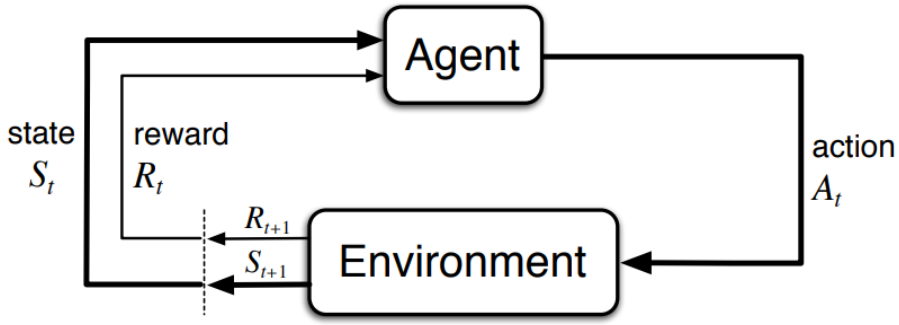


FIG. v A typical RL setup: the Agent interacts with the Environment by performing a set of actions. In response, a reward signal is produced, and a new state is observed.

At the core of RL lies the interplay between an agent and its environment: for the scope of this project, we are interested in episodic interactions. Each **episode** is a self-contained sequence of interactions that begins with the agent's initial state and ends when a terminal state is reached. The agent interacts with the environment by executing an **action** $a_t \in A$. The environment is modeled as a set of **states** S, starting from a state s_0, in response to the agent's action it moves to a new state s_t and produce a **reward** signal r_t. The reward quantifies the immediate benefit (or detriment) for the agent resulting from the chosen action. A **policy** π defines a mapping from a set of states to a set of actions: the goal of the agent is to learn an optimal policy π_* that select the appropriate action $\pi_*(s) = a$ in order to maximize the total reward obtained at the end of the episode.

The optimization problem of finding the optimal sequence for a metal-sheet bending task can be cast in the reinforcement learning domain:

- The Environment is defined as a model of the machinery (press, punch and die) in the operating configuration, along with the model of the sheet of metal and the sets of bends to perform.

- The set of States is defined as the possible intermediate configuration of the workpiece. In the simplest case, where the left/right positioning is ignored, a piece with n bends will define just 2^n states.

- The sect of actions is defined as the next bend to be performed.

The modeling of the reward signal is likely the most important part of the modeling process, and requires additional considerations. Without additional knowledge, we can assume that a *feasible* action (workpiece does not collide with the machinery during the bending action), should yield a constant positive reward, conversely in the case of a collision a strong negative reward should be returned.

$$R(s_t, a) = \begin{cases} +10.0 & \text{if action } a \text{ is feasible} \\ -1000 & \text{otherwise} \end{cases} \tag{1}$$

A more realistic reward function should enforce all the optimality criteria useful to determine the *optimal* bending sequence. Optimality criteria can be enforced for the final product (dimensional accuracy) and for the manufacturing process (throughput of the process and efficiency of implementation).

Engineering of the reward function is hence expected to take a significant effort of research and experimentation, an objective evaluation criterion of the proposed bending sequences would be highly desirable.

The state-transition must provide for the above options: a colliding action could result in the agent remaining in the same state (Eq. 2), however this option could lead to poor exploration of the state-space for methods that explore the environment interactively.

$$S\left(s_t, a\right) = \begin{cases} s_{t+1} & \text{if action } a \text{ is feasible} \\ s_t & \text{otherwise} \end{cases} \tag{2}$$

In addition, we should account for the possibility of the agent being stuck on a dead end (no successive states are reachable). This could either decree the end of the current episode, or we could allow actions that revert to previous states (undo the last bend). In the end, the exact modeling of the problems is inherently dependent on the family of approaches being considered among the broad domain of RL.

II
OPERATIVE PROPOSALS

The problem under study is defined with discrete and finite set of actions A and states S, and the dynamics of the environment can explicitly model (hence, arbitrary (s, a) pairs can be evaluated when desired).

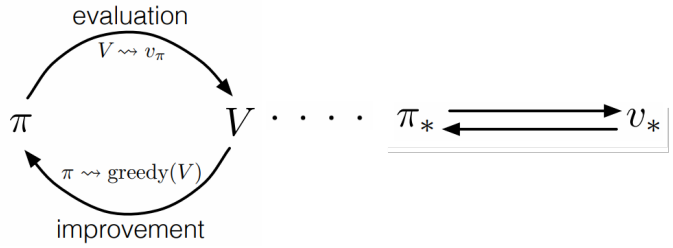


FIG. vi Iterative process of alternate phases of evaluation of the current policy and improvement.

We define the **value function** $V_\pi\left(s\right)$ of a state s under a policy π as the expected return from s following π.

$$V_\pi\left(s\right) = \mathbf{E}\{R_t | S = s_t\} \tag{3}$$

In a discrete case, the value function is a matrix $V_\pi : (S \times A) \to \mathbf{R}$ that maps the (s, a) pair to the expected reward value. Clearly, if V_π would be available, the optimal policy π_* would be defined as being *greedy* with respect to V_π, which translates to always selecting the action a that maximizes the expected reward. The core of the RL problem is hence to devise an algorithm to effectively estimate the value function.

Since in our case a perfect model of the environment is available, which is rarely the case in most RL problems, we could leverage a set of RL techniques that stems from the Dynamic Programming (DP) domain. Applying finite-state DP methods, it is possible to estimate the optimal policy π by directly estimating the value function v_π iteratively. Starting from an initial policy π_0 and evaluating its value function, v_{π_0} an improved policy π_1 can be obtained. This process, known as *policy iteration*, is guaranteed to converge to the optimal policy, however, this kind of approach suffer from a large computational complexity when dealing with a large state space. The cost of exploring and updating the value function grow exponentially, which can lead to impractical training times and memory constraints.

From the ongoing experimental evaluation, we found that Temporal Difference (TD) methods like Q-Learning could be a well suited approach. The main difference from DP methods is that TD methods update the current estimate of the value function by sampling from the environment, without requiring a complete sweep through all possible state-action pairs, making it a computationally advantageous approach.

In particular, Q-Learning learns an Action-Value function $Q \in (S \times A)$. At each step of an episode, the algorithm sample the next state with an ϵ-greedy policy, observing the reward r_t and the next state s_{t+1}, then update the current estimate for $Q(s, a)$ towards the update estimate for the *expected* reward. The latter is obtained as the sum of the *observed* reward r and the current estimate from the cumulative reward starting from the next state. This is formalized as follows:

$$Q(s,a) = Q(s,a) + \alpha \left[r_t + \gamma \left(Q\left(s', a'\right) - Q(s,a) \right) \right]$$
$$\text{with } a' = max_a Q\left(s'\right)$$

So far, we have managed to employ Q-Learning to find feasible (non-colliding) sequences with very complex workpieces (up to 17 bends). The most important findings of the preliminary experiments are:

- The importance of issuing a special reward at the end of the episode, in order to signal the agent the "winning" or "loosing" of the episode.
- The definition of the state-transition function: we found beneficial to let the agent advance to the next state even if a collision is encountered, while still issuing a negative reward. This solves the issue of occurring in "dead ends", while also promoting the exploration of later states, that would be otherwise rarely observed.
- The importance of a careful turning of the hyperparameters (the learning rate α and the discount factor γ), the values of the issued rewards, and the initialization of Q values.

Building on this early results, we plan to further assess the abilities of RL algorithms to solve our problem. A next logical step would be to embed optimality constrains in the reward function and try to obtain *optimal* sequences instead of just *feasible* ones. An orthogonal direction of work could be a biased initialization of the Q function by incorporating some rule of thumb (i.e, execute the small bends first...) to improve the ease of convergence, or even thin out the solutions space by enforcing hard-reject criteria.

Bibliography

[1] Andrea Baraldo, Luca Bascetta, Fabrizio Caprotti, Sumit Chourasiya, Gianni Ferretti, Angelo Ponti, and Basak Sakcak. Automatic computation of bending sequences for wire bending machines. *International Journal of Computer Integrated Manufacturing*, 35(12):1335–1351, 2022.

[2] B.T. Cheok, J.Y. Li, and Andrew Nee. Integrated feature-based modelling and process planning of bending operations in progressive die design. *International Journal of Advanced Manufacturing Technology*, 20:883–895, 11 2002.

[3] Joost Duflou, Jean-Pierre Kruth, and Dirk Van Oudheusden. A tsp-based algorithm for the bend sequencing problem. In *Proceedings of the 8th International Conference on Sheet Metal*, pages 69–78, 2000.

[4] Joost R Duflou, József Váncza, and Richard Aerens. Computer aided process planning for sheet metal bending: A state of the art. *Computers in industry*, 56(7):747–771, 2005.

[5] Zahid Faraz, Syed Waheed ul Haq, Liaqat Ali, Khalid Mahmood, Wasim Akram Tarar, Aamer Ahmed Baqai, Mushtaq Khan, and Syed Husain Imran. Sheet-metal bend sequence planning subjected to process and material variations. *The International Journal of Advanced Manufacturing Technology*, 88(1-4):815–826, 2017.

[6] Pedro Fernández, Braulio José Alvarez, David Blanco Fernández, Eduardo Cuesta, and Sabino Mateos. Development of a virtual machine for the sheet metal bending process simulation for educational purposes. In *Materials Science Forum*, volume 692, pages 16–23. Trans Tech Publ, 2011.

[7] Mitsuo Gen, Runwei Cheng, and Shumuel S Oren. Network design techniques using adapted genetic algorithms. *Advances in Engineering Software*, 32(9):731–744, 2001.

[8] Satyandra K Gupta, David A Bourne, KH Kim, and SS Krishnan. Automated process planning for sheet metal bending operations. *Journal of Manufacturing Systems*, 17(5):338–360, 1998.

[9] Nikolaos Kontolatis and G-C Vosniakos. Optimisation of press-brake bending operations in 3d space. *Journal of Intelligent Manufacturing*, 23:457–469, 2012.

[10] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[11] Zhang Lichao, Zhang Yi, Zhou Qiang, and He Fafu. Robust sheet metal bend sequencing method based on a-star algorithm. In *2011 IEEE International Conference on Computer Science and Automation Engineering*, volume 2, pages 711–715. IEEE, 2011.

[12] Alan C Lin and Chao-Fan Chen. Sequence planning and tool selection for bending processes of 2.5 d sheet metals. *Advances in Mechanical Engineering*, 6:204930, 2014.

[13] András Márkus, József Váncza, and András Kovács. Constraint-based process planning in sheet metal bending. *CIRP Annals*, 51(1):425–428, 2002.

[14] David R Morrison, Sheldon H Jacobson, Jason J Sauppe, and Edward C Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[15] Nils J Nilsson. *Principles of artificial intelligence*. Springer Science & Business Media, 1982.

[16] D Raj Prasanth and MS Shunmugam. Geometry-based bend feasibility matrix for bend sequence planning of sheet metal parts. *International Journal of Computer Integrated Manufacturing*, 33(5):515–530, 2020.

[17] JC Rico, JM Gonzalez, S Mateos, E Cuesta, and G Valino. Automatic determination of bending sequences for sheet metal parts with parallel bends. *International Journal of Production Research*, 41(14):3273–3299, 2003.

[18] JC Rico, S Mateos, G Valiño, CM Suárez, and E Cuesta. A methodology for the sequencing of sheet metal bending operations.

[19] AA Salem. Automatic tool selection in v-bending processes by using an intelligent collision detection algorithm. In *IOP Conference Series: Materials Science and Engineering*, volume 239, page 012002. IOP Publishing, 2017.

[20] Moshe Shpitalni and D Saddan. Automatic determination of bending sequence in sheet metal products. *CIRP annals*, 43(1):23–26, 1994.

[21] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[22] Chitra Malini Thanapandi, Aranya Walairacht, and Shigeyuki Ohara. Genetic algorithm for bending process in sheet metal industry. In *Canadian Conference on Electrical and Computer Engineering 2001. Conference Proceedings (Cat. No. 01TH8555)*, volume 2, pages 957–962. IEEE, 2001.

[23] Fedor Yu Trapeznikov and Alexandr A Petunin. A variant of solving the optimization problem of finding the order of bending of sheet metal parts with parallel bends in time. In *CEUR Workshop Proceedings*, volume 2843. CEUR-WS, 2021.

[24] Cheng-Hua Wang and David A Bourne. Using features and their constraints to aid process planning of sheet metal parts. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 1, pages 1020–1026. IEEE, 1995.

[25] Fengyu Xu, Dawei Ding, Baojie Fan, and Sen Yang. Prediction of bending parameters and automated operation planning for sheet-metal bending orientated to graphical programming. *The International Journal of Advanced Manufacturing Technology*, 126(5-6):2191–2204, 2023.