Metaheuristics and Supervisory Control for Flexible Manufacturing System Production Planning with HMM-Concat

Ana Clara P. Goncalves
Electrical Engineering Graduate Program
Federal University of Minas Gerais
Belo Horizonte, MG, Brazil
anacpg@ufmg.br

Patrícia N. Pena
Department of Electronic Engineering
Federal University of Minas Gerais
Belo Horizonte, MG, Brazil
ppena@ufmg.br

Daniel Sarsur
Electrical Engineering Graduate Program
Federal University of Minas Gerais
Belo Horizonte, MG, Brazil
danielsc@ufmg.br

***Abstract*—This paper proposes a solution, named HMM-concat, which combines the Supervisory Control and Optimization (SCO) methodology with the Heuristic Makespan Minimization (HMM) methodology to address the challenging task of production planning in Flexible Manufacturing Systems (FMS). The approach integrates metaheuristic optimization techniques and supervisory control methods for discrete event dynamic systems, aiming to minimize the makespan for large production batches. This paper utilized the Flexible Manufacturing System (FMS) implemented and located at the Laboratory for Analysis and Control of Discrete Event Systems (LACSED) as the problem to be modeled and solved, employing the UltraDES and PlanningDES libraries. Empirical validation demonstrated that HMM-concat consistently achieves optimal makespan and significantly improves execution times for larger batch sizes compared to the standalone HMM algorithm. For instance, for a (1000, 1000) batch size, HMM-concat completed the task in 10.336 seconds while HMM took 140.278 seconds, highlighting a performance improvement.**

***Index Terms*—Discrete Event Systems; Metaheuristics; Supervisory Control Theory; UltraDES; Automata.**

I. Introduction

Traditional manufacturing processes have significantly advanced with the introduction of Flexible Manufacturing Systems (FMS) as a replacement for conventional production systems [1]. The adoption of these systems in manufacturing environments offers advantages, mainly due to their greater adaptability to market changes [1]. These systems face high complexity and high operational costs. Therefore, there is a continuous effort to automate and optimize these systems, seeking better performance [2].

The challenge of sequencing tasks in Flexible Manufacturing Systems is crucial in the modern industrial context. Optimizing performance is fundamental to increasing competitiveness. This critical step not only affects operational efficiency but also the utilization of available resources [3]. Metaheuristics, with their diverse approaches and adaptations, represent a research and development area with great potential to significantly contribute to this optimization. While some techniques may be more suitable for smaller-scale problems, others can be expanded to face more complex challenges, always with the aim of finding efficient and applicable solutions in an industrial context [7].

This paper addresses the critical challenge of task sequencing in Flexible Manufacturing Systems (FMS) with the objective of makespan minimization for large production batches. This problem is particularly relevant in dynamic industrial environments where efficient resource utilization and timely product delivery are important. Our work builds upon existing research in this area, particularly the Supervisory Control and Optimization (SCO) methodology [2] and the Heuristic Makespan Minimization (HMM) approach [3]. The SCO methodology combines metaheuristic optimization and supervisory control methods for discrete event dynamic systems. The HMM algorithm can face high execution times for complex instances [3].

This paper proposes the implementation of a new solution, named HMM-concat, which combines elements from the SCO methodology (originally introduced by Costa et al., 2018 [2]) with the HMM methodology (developed by Alves et al., 2021 [3]). This integration leverages the strengths of both metaheuristic optimization and supervisory control theory to achieve efficient production planning for larger production volumes.

The main contributions of this work are:

- The development and implementation of the HMM-concat algorithm, an approach that combines metaheuristic search strategies (specifically VNS and 2-opt) with supervisory control theory to optimize task sequencing in FMS, aiming to minimize the makespan.
- Demonstration of the HMM-concat algorithm's ability to achieve optimal makespan for various production batch sizes, with significantly reduced execution times for larger batches compared to the standalone HMM algorithm.

For the implementation of the solution, tools developed at the Laboratory for Analysis and Control of Discrete Event

This work is partially funded by the agencies Coordenação e Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 and Fapemig P/APQ-06580-24.

Systems (LACSED) were utilized, such as UltraDES and PlanningDES [8].

The present paper is structured as follows. The current section provides an introduction to the problem and the paper's contributions. Section 2 presents preliminary notions of Discrete Event Systems, Supervisory Control Theory, and Metaheuristics. In Section 3, the description of the implemented HMM-concat algorithm is detailed. Section 4 demonstrates the practical results of the algorithm compared to other algorithms. Finally, Section 5 presents the conclusions of this article and outlines avenues for future work.

II. Preliminary Concepts

A. Discrete Event Systems

An alphabet Σ is a finite set of symbols. A word s (or string) is a finite sequence of symbols from an alphabet Σ. The empty word is denoted by e. A language L is a subset of Σ^*, and $\overline{L}$ represents the set of all prefixes of strings in L. The Kleene closure, Σ^*, encompasses all finite-length words formed from Σ, including the empty word [5].

A Deterministic Finite Automaton (DFA) is formally defined as $G = (Q, \Sigma, \delta, q_0, Q_m)$ [5], where:

- Q is a finite set of states;
- Σ is the alphabet;
- $\delta : Q \times \Sigma \longrightarrow Q$ is the state transition function;
- $q_0 \in Q$ is the initial state;
- $Q_m \subseteq Q$ is the set of marked states.

An automaton G generates a language $\mathcal{L}(G)$ (all possible sequences of events from q_0) and marks a language $\mathcal{L}_m(G)$ (sequences leading to marked states) [5].

B. Supervisory Control Theory

In 1987, Ramadge and Wonham developed the mathematical framework for Supervisory Control Theory (SCT), which became instrumental in the systematic analysis and design of controllers for discrete event systems [4].

SCT is a formal framework for regulating the behavior of Discrete Event Systems (DES), ensuring safe and efficient operation according to predefined specifications [6]. The system's uncontrolled behavior is modeled by a finite automaton called the plant, which describes its physical capabilities and all possible behaviors [6].

Events are classified as controllable (Σ_c), which can be disabled by the supervisor, or uncontrollable (Σ_u), which cannot be disabled [6]. The goal is to synthesize a supervisor such that the closed-loop system is non-blocking (i.e., can always reach its marked states) and its language K is controllable, meaning that any uncontrollable event that can occur in the plant must also be allowed by the supervisor within the desired behavior [6].

C. Heuristics and Metaheuristics

Heuristics and metaheuristics are problem-solving techniques used primarily for optimization problems where exact methods are computationally prohibitive. Heuristics offer practical and efficient approaches to find reasonable solutions, often guided by simple rules or experience, though optimality is not guaranteed. Metaheuristics, conversely, are higher-level strategies designed to explore the solution space more broadly, aiming for high-quality solutions and adaptability across a wide range of optimization problems [7].

This paper employs two metaheuristic algorithms: Variable Neighborhood Search (VNS) and 2-opt swap. VNS systematically explores a variety of predefined neighborhoods to avoid getting trapped in local optima and find better solutions [13]. The 2-opt swap is a local search technique widely applied in permutation problems, which improves a solution by reversing a segment of the sequence [13].

D. Parallelism Maximization Algorithms

Parallelism maximization algorithms were utilized for generating initial sequences and for comparative analysis due to their strong performance and compatibility with UltraDES and PlanningDES libraries [8], [10]. The Heuristic Makespan Minimization (HMM) algorithm, though non-polynomial in complexity, significantly reduces the search space by prioritizing controllable events over uncontrollable ones [3]. Furthermore, HMM prunes search paths by retaining only the shortest-time path when multiple paths lead to the same state [3]. While this can still lead to high execution times for highly complex instances, specific heuristics within HMM effectively manage the branching factor, making it valuable for obtaining efficient initial sequences [3].

E. Flexible Manufacturing System

The proposed solution is validated through a case study involving a Flexible Manufacturing System (FMS) prototype, represented by Figure 1, first described in [9] and currently implemented at the LACSED laboratory [11]. This FMS is well-suited for validation as its optimal makespans are known, facilitating result comparison [11].

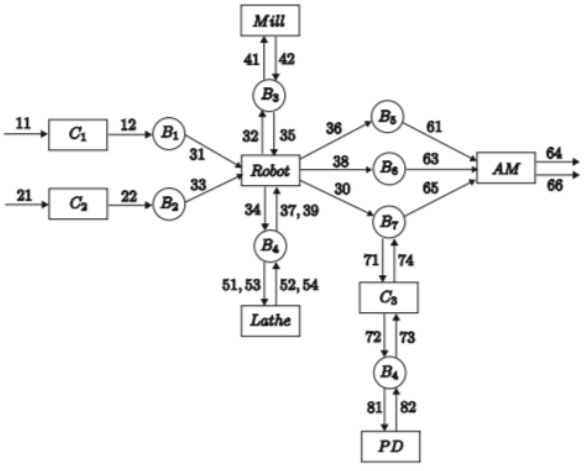

Fig. 1. Sistema flexível de manufatura *Fonte: [9]*

The system processes two types of parts (blocks and billets) to produce two distinct products: product A (conical pin top) and product B (cylindrical painted pin) [9]. The FMS

comprises eight interconnected pieces of equipment: three conveyors (C_1, C_2, and C_3), a mill, a lathe, a robot, a painting device (PD), and an assembly machine (AM) [9]. The flow of parts and operations within the FMS is represented by events (indicated by arrows in Figure 1), with parts moving between equipment via buffers (B_1 to B_8), each capable of storing one unit [9]. Events such as part loading, machine processing, and robot transfers are considered controllable, while events indicating machine completion or sensor triggers are typically uncontrollable, impacting how sequences are managed by the supervisor.

III. The HMM-Concat Algorithm

The fabrication of each product is described by a specific sequence of events. We assume that a command sequence of length n, denoted as $\epsilon = e_1, e_2, e_3, ..., e_{n-1}, e_n$, results in a product P in a plant.

Some of these subsequences, represented within ϵ, must be executed in a specific order. Failing to maintain this relative order would prevent the product from being manufactured, as the system could either reach a forbidden state or attempt an operation that is infeasible within the plant's operational constraints if not properly supervised. These are termed invariant subsequences. However, the order of events between different invariant subsequences can be altered to manufacture product P, provided that the internal relative order of elements within each invariant subsequence is preserved.

In the Flexible Manufacturing System (FMS) (Figure 1), for each product (A and B), there are three invariant subsequences: one for manufacturing the base and two for the respective pins (one for product A and one for product B). The necessity of maintaining a fixed internal ordering for these subsequences is directly related to operations that have a natural precedence (e.g., the pin must be turned before it is painted).

Under certain conditions and without loss of generality, optimization can be applied solely to the sequences of controllable commands (Σ_c^*) [12]. Considering the controllable events of the FMS, the invariant subsequences are encoded as follows to facilitate manipulation by the optimization algorithms:

- The sequence for manufacturing the base of product A:
$$\epsilon_1 = \{11, 31, 41, 35, 61\} \longrightarrow a_0a_0a_0a_0a_0$$
- The sequence for manufacturing the pin of product A:
$$\epsilon_2 = \{21, 33, 51, 37, 63\} \longrightarrow a_1a_1a_1a_1a_1$$
- The sequence for manufacturing the base of product B has the same events as product A's base:
$$\epsilon_3 = \{11, 31, 41, 35, 61\} \longrightarrow b_0b_0b_0b_0b_0$$
- The sequence for manufacturing a cylindrical pin for product B:
$$\epsilon_4 = \{21, 33, 53, 39, 71, 81, 73, 65\} \longrightarrow b_1b_1b_1b_1b_1b_1b_1b_1$$

Thus, to produce one unit of product A and one unit of product B, these symbolic event sequences must be concatenated. For instance, a base sequence for two products would combine these symbolic strings. For illustration, a simplified sequence representing the production of one product (A or B), denoted as p, could be:

$$p = a_0a_0a_0a_0a_0a_1a_1a_1a_1a_1b_0b_0b_0b_0b_0b_1b_1b_1b_1b_1b_1b_1$$

The SCO algorithm [2] applies Variable Neighborhood Search (VNS) and generates new candidate sequences for evaluation. The shuffling of elements within these sequences, through the generation of new neighborhood structures, allows for a more comprehensive exploration of the search space, increasing the chances of finding high-quality solutions for the problem.

The SCO-concat approach [2] is designed to process larger production batches. Its objective is to find an efficient way to manufacture different production lots. This is achieved by first concatenating sequences previously optimized for smaller lots. The resulting concatenated sequence is then optimized using VNS and 2-opt swap, and evaluated by a function referred to as the Legal Sequence Generator and Evaluator (LSGE).

The HMM-concat approach is an alternative version of the SCO-concat, drawing elements from [2]. The objective of HMM-concat is also to find an efficient way to manufacture different production lots in parallel. In this approach, sequences previously optimized by the HMM algorithm [3] for smaller lots are initially concatenated. The resulting sequence is then optimized using VNS and evaluated by a function to verify the sequence's feasibility.

For HMM-concat, the process begins by using the result of a single product A and a single product B batch, obtained from the HMM algorithm. This initial sequence consists of 23 controllable events. For illustrative purposes, this sequence can be generalized as:

$$p = a_0a_0a_1a_1b_0b_0b_1,$$

which produces one product (where a_0, a_1, b_0, b_1 represents the symbolic invariant subsequences described earlier).

A solution for producing two products would be the concatenation of two such sequences. To manage these concatenated sequences, a masking mechanism is introduced, represented as:

$$p_{concat} = \underbrace{a_0\ a_0\ a_1\ a_1\ b_0\ b_0\ b_1}_{\text{mask 0}}\ \overbrace{a_0\ a_0\ a_1\ a_1\ b_0\ b_0\ b_1}^{\text{mask 1}},$$

This combined sequence represents the production of two products.

This concatenated sequence can be conceptually divided into three phases:

1) Initial Phase: Machines involved in the first stages begin operation, while other parts of the system may still be idle.
2) Production Phase: The majority of the manufacturing processes occur, typically representing the steady-state production.
3) Final Phase: The system completes the current batch, and machines transition back to an idle state.

$$p_{concat} = \underbrace{0\,0\,0\,0}_{\text{initial phase}}\ \overbrace{0\,0\,0\,1\,1\,1\,1}^{\text{production phase}}\ \underbrace{1\,1\,1}_{\text{final phase}}$$

It is important to note that the sum of the number of events in the initial and final phases must equal the sum of the number of events in the production phase, ensuring the concatenated sequence has the correct total number of events. The optimization algorithms (VNS and 2-opt swap) are primarily applied to the production phase, from which a pattern (the sequence of events generated by VNS) is created. This strategy aims to find the optimal sequencing for the intermediate part of the production process, where the system is most intensively utilized.

In VNS, local search is iteratively performed in different neighborhoods of the current solution. A neighborhood is defined as a set of solutions reachable from the current solution through a single type of move. VNS creates a new solution by applying the 2-opt swap method to shuffle events within the production phase. It then checks if the makespan of this new solution is smaller than that of the current solution. If it is, the current solution is updated; otherwise, the algorithm moves to the next neighborhood. This approach leverages different neighborhood structures to increase the chance of finding higher-quality solutions.

$$p_{concat} = 0\,0\,0\,0\ \overbrace{1\,1\,0\,0\,1\,0\,1}^{\text{pattern}}\ 1\,1\,1$$

Following this, an ipattern (inverted pattern) is generated by swapping the values of the masks (e.g., if $pattern = 01100101$, then $ipattern = 10011010$). To expand a sequence adequately for multiple products, the pattern and ipattern are alternated. This concatenation process follows a specific logic based on the number of products, ensuring an efficiently organized resulting sequence. For an odd number of products, the sequence terminates with an ipattern; for an even number, it ends with a pattern. Additionally, the final phase also varies with the number of products: odd numbers of products end with a mask of 0s, while even numbers end with a mask of 1s. This method of alternating patterns ensures that the expanded sequence is organized to maximize processing efficiency, assuming the existence of optimal or sub-optimal cycles. An example for the production of 3 products should be:

$$p_{concat} = \underbrace{0\,0\,0\,0}_{\text{initial phase}}\ \overbrace{1\,1\,0\,0\,1\,0\,1}^{\text{pattern}}\ \underbrace{0\,0\,1\,1\,0\,1\,0}_{\text{ipattern}}\ \overbrace{1\,1\,1}^{\text{final phase}},$$

A common challenge with this shuffling is the potential generation of infeasible sequences. This is addressed using the Legal Sequence Generator and Evaluator (LSGE) function. If an infeasible solution is detected, LSGE applies a procedure that corrects the position of events, ensuring they conform to a feasible behavior. The LSGE algorithm specifically verifies the transitions of the finite automaton associated with the FMS to ensure compliance with the closed-loop model. If an event does not have an immediate available transition, it is postponed and periodically re-evaluated to maintain processing continuity.

A flowchart of this algorithm is represented on Figure 2.

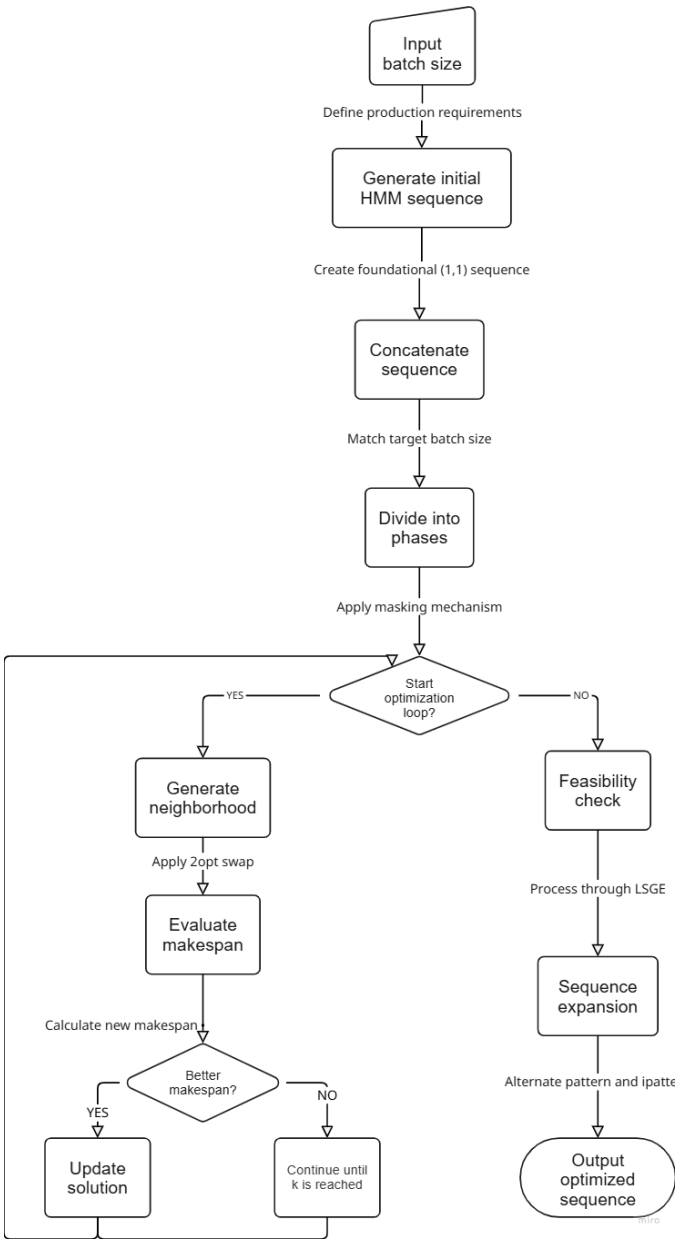

Fig. 2. Flowchart of the aplication HMM-concat algorithm.

IV. Results

The experiments were conducted on a computer equipped with an Intel Core i9-9900K/3.60 GHz processor and 64 GB of RAM. Tests were performed using three distinct algorithms:

1) The HMM algorithm [3];
2) The SCO-concat algorithm [2];
3) The HMM-concat algorithm.

A. Optimization Parameters

After generating and concatenating an initial sequence, the algorithm applies a Variable Neighborhood Search (VNS) to optimize the production phase of the sequence. The specific parameters for the VNS are:

- Number of VNS Iterations: The VNS main loop is set to run for a maximum of 10 iterations. This parameter defines the overall search effort of the metaheuristic.
- Neighborhood Size: The VNS is configured to explore 5 different neighborhood structures. In each iteration, the algorithm systematically perturbs the current solution by exploring these increasingly distant neighborhoods to escape local optima.
- Local Search Intensity: Within each neighborhood, a local search is performed. This parameter is set to 10, indicating that the 2-opt swap method is applied up to 10 times to intensively search for improvements within the current neighborhood before proceeding.

These parameters were determined empirically.

B. Comparison of HMM-concat with HMM

Table I and Figure 3 present the results of tests comparing the HMM-concat algorithm with the standalone HMM algorithm. In these tests, batch size refers to the number of product A and product B units. As shown in Table I, the makespan achieved by HMM-concat, which utilizes HMM as its initial sequence generator, is consistently optimal, matching that of the pure HMM [3].

Crucially, it is observed that from a batch size of (50,50) onwards, the execution time of HMM-concat is significantly lower when compared to HMM. This substantial reduction in execution time for larger instances highlights HMM-concat's practical scalability and efficiency. While HMM rigorously explores numerous possible execution paths to guarantee optimal makespan, its combinatorial complexity becomes prohibitive for increasing batch sizes, leading to considerably longer processing times. HMM-concat, conversely, leverages pre-optimized base sequences and applies metaheuristic search (VNS and 2-opt) primarily within the production phase of the concatenated sequence, effectively reducing the exhaustive search space and computational burden, especially for large production volumes. This practical efficiency is a key advantage for real-world industrial applications.

C. Comparison of SCO, HMM-Concat, and SCO-Concat

Table II and Figure 4 illustrate the makespan values for different batch sizes using the SCO algorithm [2], HMM-concat, and SCO-concat [2]. HMM-concat's initial sequence is made by HMM with a (1,1) product batch, and SCO-concat's initial sequence is made by SCO with a (1,1) product batch [2]. These tests highlight that the quality of the initial sequence is of paramount importance for the algorithm to consistently achieve the optimal makespan within efficient execution times across all tested batch sizes.

TABLE I
MAKESPAN AND EXECUTION TIME OF HMM [3] AND HMM-CONCAT ALGORITHMS WITH OPTIMAL INITIAL SEQUENCE

Batch Size	HMM		HMM-Concat	
	Makespan	Time (s)	Makespan	Time (s)
(1,1)	238	0.052	238	0.051
(5,5)	866	0.278	866	2.014
(10,10)	1651	0.476	1651	1.801
(15,15)	2436	0.566	2436	2.153
(50,50)	7931	4.649	7931	3.753
(100,100)	15781	11.061	15781	2.855
(150,150)	23631	17.687	23631	3.045
(500,500)	78581	66.065	78581	5.165
(750,750)	117831	103.034	117831	9.008
(1000,1000)	157081	140.278	157081	10.336

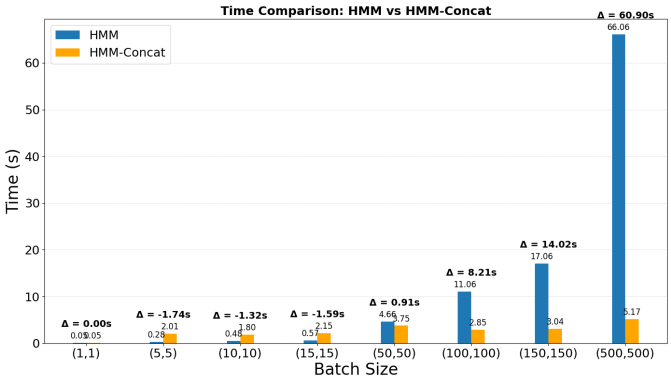

Fig. 3. Execution time difference between HMM and HMM-concat.

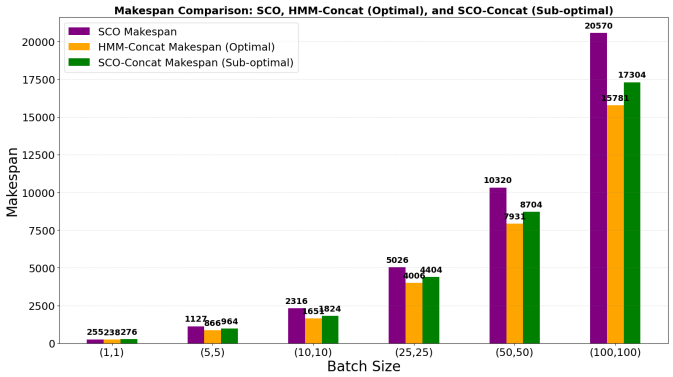

Fig. 4. Makespan difference between SCO, HMM-concat, and SCO-concat.

D. Impact of Initial Sequence Quality on HMM-Concat

Table III and Figure 5 present the makespan and execution time values for HMM-concat when initialized with sequences from (1,1), (2,2), and (3,3) product batches. It is consistently observed that the values obtained with the (1,1) initial sequence yield superior performance.

The consistent superior performance observed with the (1,1) initial sequence, both in terms of achieving optimal makespan and often faster execution, is a key finding. This suggests that the (1,1) batch provides an inherently more efficient production pattern that the metaheuristic can effectively replicate and optimize for larger scales. This efficiency likely stems from

TABLE II
RESULTS FOR SCO, HMM-CONCAT, AND SCO-CONCAT WITH (1,1) INITIAL BATCH SEQUENCE

Batch Size	SCO		HMM-concat		SCO-concat	
	Makespan	Time (s)	Makespan	Time (s)	Makespan	Time (s)
(1,1)	255	13.961	238	0.009	276	0.008
(5,5)	1127	17.149	866	1.864	964	13.469
(10,10)	2316	22.244	1651	2.036	1824	11.065
(25,25)	5026	63.959	4006	1.938	4404	13.693
(50,50)	10320	159.857	7931	2.157	8704	13.587
(100,100)	20570	353.859	15781	3.906	17304	17.830

its minimized idle times and maximized resource utilization during the fundamental production cycle. Furthermore, using a smaller base sequence like (1,1) offers greater adaptability for dynamic adjustments within the Legal Sequence Generator and Evaluator (LSGE) algorithm, allowing for efficient correction of infeasible event positions without significantly increasing computational overhead.

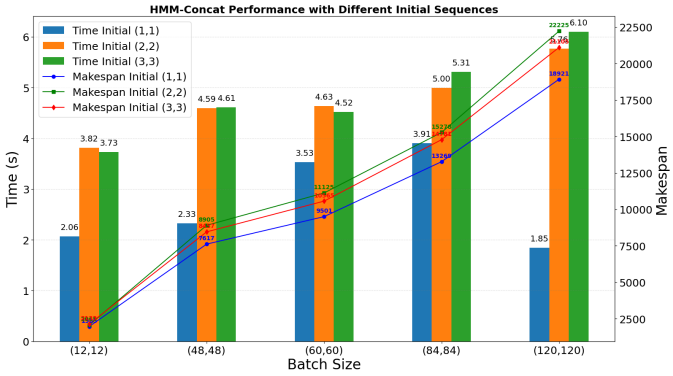

Fig. 5. Execution time and makespan difference of HMM-concat with different initial sequences.

E. Algorithmic Complexity Discussion

While a formal computational complexity analysis (e.g., in terms of Big O notation) for metaheuristic algorithms like HMM-concat can be challenging due to their iterative and stochastic nature, we can discuss the practical implications of their complexity. The foundational Heuristic Makespan Minimization (HMM) algorithm, utilized for generating initial sequences, is characterized by its non-polynomial complexity. This inherent complexity arises from the combinatorial explosion of states and event sequences in Discrete Event Systems (DES), leading to a rapid increase in computation time as problem size (batch size) grows. Even with pruning heuristics, HMM can become computationally prohibitive for large instances.

The HMM-concat algorithm addresses this challenge by employing Variable Neighborhood Search (VNS) and 2-opt swap within its optimization phase. Metaheuristics, by design, do not guarantee optimal solutions in polynomial time, but they aim to find high-quality solutions within practical time limits. The complexity of HMM-concat's metaheuristic search depends on factors such as the number of VNS iterations, the size of the explored neighborhoods, and the complexity of the 2-opt local search (which can be O(N^2) for a sequence of length N). Each candidate sequence generated by the metaheuristic must also be evaluated for its makespan and corrected for feasibility by the Legal Sequence Generator and Evaluator (LSGE) function. This evaluation step involves a simulation of the FMS, adding to the per-iteration computational cost.

Despite these complexities, the empirical results demonstrate HMM-concat's superior practical efficiency for large batches. The significant reduction in execution time observed for HMM-concat compared to the standalone HMM (e.g., for 1000 product batches) shows that our metaheuristic approach effectively navigates and optimizes the large combinatorial search space, circumventing the exponential growth that limits HMM for larger instances. This practical scalability, evidenced by the improved wall-clock execution times, highlights HMM-concat's utility for industrial-scale production planning.

V. CONCLUSION

The primary contribution of this article was the development of the HMM-concat algorithm, leveraging the capabilities of libraries developed by LACSED, with significant input from PlanningDES [8].

This work allowed for the re-implementation and comparison of these algorithms, including those presented in [2], with newer algorithms developed within the research group [10]. This comparison enables an assessment of HMM-concat's performance in a contemporary context, identifying its strengths and potential areas for improvement [10]. Among its strengths, the capacity to find high-quality solutions for complex sequencing problems stands out. In summary, the obtained results demonstrate that the HMM-concat algorithm is a robust and efficient solution for the production planning problem in Flexible Manufacturing Systems.

Certain limitations should be noted. The experiments were conducted on a high-performance computer with an Intel Core i9 processor and 64 GB of RAM. The memory and processing requirements, especially for the underlying automaton model and the concatenated event sequences, may scale with very large batch sizes. Consequently, its application on hardware with more constrained resources, such as embedded controllers commonly found in industrial settings, has not been evaluated and may pose a challenge. This highlights a gap between the current validation and deployment in less robust hardware environments.

TABLE III
RESULTS - HMM-CONCAT WITH (1,1), (2,2), AND (3,3) INITIAL BATCH SEQUENCES

Batch Size	Initial (1,1)		Initial (2,2)		Initial (3,3)	
	Makespan	Time (s)	Makespan	Time (s)	Makespan	Time (s)
(12,12)	1965	2.065	2065	3.816	2133	3.732
(48,48)	7617	2.327	8905	4.591	8457	4.614
(60,60)	9501	3.533	11125	4.635	10565	4.519
(84,84)	13269	3.907	15278	4.997	14781	5.308
(120,120)	18921	1.847	22225	5.764	21105	6.099

Another restraint is that the current implementation and all presented experiments are constrained to symmetrical production batches, where the number of units for Product A is equal to the number of units for Product B (e.g., 10 of A and 10 of B). The algorithm, in its current form, has not been designed or validated for asymmetrical production scenarios (e.g., 10 of A and 15 of B), which are common in real-world manufacturing.

A. Future Work

Future research directions can explore several limitations of this work. While HMM-concat demonstrates efficiency in minimizing the makespan, some limitations must be considered. First, our validation and comparison were conducted using a specific Flexible Manufacturing System at the LACSED laboratory, utilizing our libraries UltraDES and PlanningDES. This choice facilitated the comparison with other algorithms developed by our research group, which rely on these libraries. However, future work should include comparison with other research groups to broaden the scope of evaluation. As well as expanding the algorithm to fit asymmetrical production scenarios.

Another area to explore is the application of the HMM-concat algorithm to other types of manufacturing systems or to more complex FMS configurations, potentially incorporating additional real-world constraints such as machine failures, maintenance schedules, or dynamic resource availability.

Additionally, a more in-depth analysis of different metaheuristic parameter tunings could potentially enhance the algorithm's performance. Finally, exploring adaptive initial sequence generation strategies that can dynamically respond to varying production demands or system states could significantly improve the algorithm's versatility and applicability in highly dynamic industrial environments.

REFERENCES

[1] W. L. Simões, R. Dalla Vecchia, and M. G. DaSilva, "Proposição de um modelo de otimização para programação da produção em Sistema Flexível de Manufatura (FMS) com tempos de setup dependentes da sequência: a combinação de esforços em sequenciamento e tempos de preparação na indústria eletrônica," Produto & Produção, vol. 16, no. 1, 2015.

[2] T. A. Costa, P. N. Pena, and R. H. Takahashi, "Sco-concat: A Solution to a Planning Problem in Flexible Manufacturing Systems Using Supervisory Control Theory and Optimization Techniques," J. Control, Autom. Elect. Syst., vol. 29, pp. 500–511, 2018.

[3] L. V. R. Alves, P. N. Pena, and R. H. C. Takahashi, "Planning on Discrete Event Systems using parallelism maximization," Control Eng. Pract., vol. 112, p. 104813, 2021.

[4] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event systems," *SIAM J. Control Optim.*, vol. 25, no. 5, pp. 1202–1218, 1987.

[5] C. G. Cassandras and S. Lafortune, Introduction to Discrete Event Systems. New York, NY, USA: Springer, 2008.

[6] W. M. Wonham, K. Cai, and K. Rudie, "Supervisory control of discrete-event systems: A brief history," *Annu. Rev. Control*, vol. 45, pp. 250–256, 2018.

[7] E.-G. Talbi, Metaheuristics: From Design to Implementation. Hoboken, NJ, USA: Wiley, 2009.

[8] L. V. R. Alves, L. R. R. Martins, and P. N. Pena, "UltraDES - A Library for Modeling, Analysis and Control of Discrete Event Systems," in 20th IFAC World Congress, Toulouse, France, Jul. 2017, vol. 50, no. 1, pp. 5831–5836.

[9] M. H. de Queiroz, J. E. R. Cury, and W. M. Wonham, "Multitasking supervisory control of discrete-event systems," Discrete Event Dyn. Syst., vol. 15, pp. 375–395, 2005.

[10] L. V. R. Alves, G. C. Rafael, L. S. Batista, and P. N. Pena, "A multi-objective approach for manufacturing systems with multiple production routes based on supervisory control theory and heuristic algorithms," Discrete Event Dyn. Syst., vol. 33, no. 4, pp. 373–394, 2023.

[11] R. Malik and P. N. Pena, "Optimal task scheduling in a flexible manufacturing system using model checking," in 7th IFAC Conference on Manufacturing Modelling, Management and Control, Bratislava, Slovakia, Jun. 2018, vol. 51, no. 7, pp. 230–235.

[12] P. N. Pena, J. N. Vilela, M. R. C. Alves, and G. C. Rafael, "Abstraction of the Supervisory Control Solution to Deal With Planning Problems in Manufacturing Systems," IEEE Trans. Autom. Control, vol. 67, no. 1, pp. 344–350, Jan. 2022.

[13] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *Eur. J. Oper. Res.*, vol. 130, pp. 449–467, Feb. 2001.