A Hybrid Scatter Search Algorithm for the Sequencing m-Vector Bin Packing Problem

Isma Dahmani[1], Mhand Hifi[2,⋆], and Khadidja Latrem[3]

***Abstract*— his paper addresses the Sequencing m-Vector Bin Packing Problem (m-BiPacS), an NP-hard challenge requiring multidimensional capacity and job-sequencing decisions. We propose a *cooperative scatter search algorithm* combining structured reference-set search with a warm-started IBM ILOG CPLEX mixed-integer programming repair phase on restricted subproblems. On standard benchmarks, the designed method reaches the best-known solution on all 20 instances and reduces the average absolute performance gap from 15.0 for the state-of-the-art *iterative variable neighborhood heuristic* and 17.7 for *standard scatter search* to 0.2.his paper addresses the Sequencing m-Vector Bin Packing Problem (m-BiPacS), an NP-hard challenge requiring multidimensional capacity and job-sequencing decisions. We propose a *cooperative scatter search algorithm* combining structured reference-set search with a warm-started IBM ILOG CPLEX mixed-integer programming repair phase on restricted subproblems. On standard benchmarks, the designed method reaches the best-known solution on all 20 instances and reduces the average absolute performance gap from 15.0 for the state-of-the-art *iterative variable neighborhood heuristic* and 17.7 for *standard scatter search* to 0.2.T**

I. Introduction

Modern production systems face dual pressures of mass customization and cost efficiency. As demand for personalization rises, traditional assembly lines struggle to manage the resulting complexity and variability. Component supply logistics has emerged as a critical bottleneck: ensuring the right parts reach the right workstation at the right time, without disrupting the production rhythm. This shift has made the integration of packaging and sequencing processes essential, particularly through *"kitting"* strategies where components are pre-assembled into product-specific sets to streamline assembly and reduce handling times. In this context, flexible models such as *"multiple-piece-flow assembly lines"* allow multiple customized workpieces to be processed simultaneously within a shared cycle, reducing idle times and increasing throughput. However, coordinating these flows under tight resource and timing constraints constitutes a fundamentally hard combinatorial challenge. This gives rise to the Sequencing m-Vector Bin Packing Problem (m-BiPacS), sitting at the intersection of scheduling and multidimensional resource allocation. The m-BiPacS combines multidimensional bin packing with time-dependent job sequencing, requiring the assignment of jobs to discrete launch times while respecting capacity constraints and temporal requirements [15]. The m-BiPacS is NP-hard and computationally prohibitive for large-scale instances. Its intractability stems from the simultaneous interplay of multidimensional capacity constraints, release dates, and weighted assignment costs, rendering even medium-sized instances beyond the reach of exact solvers within practical time limits. While exact algorithms guarantee optimality, they lack scalability; conversely, metaheuristics offer better scalability but often suffer from premature convergence. This duality motivates hybrid approaches combining the structural precision of mathematical programming with the exploratory power of metaheuristics.

Building on [1] and [15], this paper presents an adaptive hybrid approach based on the *Scatter Search* (SS) framework [4], [8], integrating: (i) a solver-assisted diversification mechanism, (ii) reference set learning, and (iii) a combination of population-based search and mathematical programming. Unlike evolutionary algorithms relying on random variation, SS systematically diversifies and intensifies the search through structured solution combination and strategic reference set management. SS is preferred to a classical Genetic Algorithm because its memory-based, partly deterministic mechanism preserves elite and diverse feasible structures under tight capacities.

In this paper, the following abbreviations are employed: VBP, Vector Bin Packing; m-BiPacS, Sequencing m-Vector Bin Packing Problem; SS, Scatter Search; CSSA, Cooperative Scatter Search Algorithm; MIP, mixed-integer programming; IVNH, Iterative Variable Neighborhood Heuristic; AAP, average absolute performance gap; TTB, time-to-best. Computational experiments on benchmark instances derived from [15] show that the proposed approach consistently outperforms state-of-the-art methods in solution quality and efficiency, particularly for tight-deadline (SD), high-dimensional instances. The remainder of this paper is organized as follows. Section II formally defines the m-BiPacS. Section III reviews related work. Section IV describes the SS framework. Section V details the proposed method. Section VI presents computational results, and Section VII concludes.

II. Problem Definition

The m-BiPacS emerges from the operational reality of multi-piece-flow assembly systems, where multiple product variants must be simultaneously processed along shared production lines under strict resource and timing constraints.

⋆Corresponding author (M. Hifi). All authors are listed in alphabetical order of their last name.

[1]AMCD-RO, USTHB, BP 32 El Alia, 16111 Alger, Algérie (isma.dahmani.usthb@gmail.com)

[2]EPROAD UR 4669, Université de Picardie Jules Verne, 80000 Amiens, France (mhand.hifi@u-picardie.fr)

[3]LaROMaD, USTHB, BP 32 El Alia, 16111 Alger, Algérie (latremkh@gmail.com)

Such a problem combines aspects of multidimensional bin packing and job sequencing under temporal and resource constraints. The objective is to assign each workpiece to a discrete launch time—referred to as a bin—such that the overall weighted assignment cost is minimized while ensuring feasibility with respect to multidimensional capacity constraints and timing windows.

A. Notation and Parameters

Let $J = \{1, \ldots, n\}$ be the set of jobs (workpieces), and let $B = \{1, \ldots, \bar{B}\}$ be the set of discrete time bins (launch times), where $\bar{B}$ is the maximum allowable number of bins. Each job $j \in J$ is characterized by the following parameters:

- r_j: the earliest feasible launch time (release date),
- d_j: the preferred deadline for job j,
- $v_j = (v_{j1}, \ldots, v_{jm})$: a resource consumption vector across m dimensions,
- w_{jb}: the assignment cost of job j to bin b, satisfying $w_{jb} \leq w_{jb'}$ for $b \leq b'$ (i.e., cost is non-decreasing with delay).

Each bin $b \in B$ has a uniform capacity c along each resource dimension $i \in \{1, \ldots, m\}$, which must not be exceeded. Let $x_{jb} \in \{0, 1\}$ be a binary decision variable equal to 1 if job j is assigned to bin b, and 0 otherwise.

B. MIP Formulation

The mixed-integer programming (MIP) formulation of the m-BiPacS is as follows:

$$\min \quad \sum_{j \in J} \sum_{b=r_j}^{\bar{B}} w_{jb} x_{jb} \tag{1}$$

$$\text{s.t.} \quad \sum_{b=r_j}^{\bar{B}} x_{jb} = 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{j \in J} v_{ji} x_{jb} \leq c \qquad \forall i \in \{1, \ldots, m\},\ \forall b \in B \tag{3}$$

$$x_{jb} \in \{0, 1\} \qquad \forall j \in J,\ \forall b \in \{r_j, \ldots, \bar{B}\} \tag{4}$$

Equation (1) defines the objective: to minimize the total weighted assignment cost. Constraint (2) ensures each job is assigned to exactly one bin within its allowed launch window. Constraint (3) ensures that, for every dimension and every time bin, the total resource consumption does not exceed the bin capacity. Constraint (4) enforces binary decision variables. Collectively, these constraints model a tightly coupled scheduling problem where temporal feasibility and multidimensional capacity must be satisfied simultaneously — a combination that is responsible for the problem's NP-hardness [15]. The benchmark assumes independent jobs; kit-component dependencies could be added through linking or precedence constraints and handled by the same MIP repair, but are outside this study.

Illustrative Example. Figure 1 illustrates the pipeline execution of two bins B_1 and B_2 across three assembly stations. Jobs j_1 and j_3 are grouped in B_1, while j_2 and j_5 belong to B_2. At Station 2, B_2 cannot begin processing until B_1 completes, resulting in a precedence delay (red dashed line).

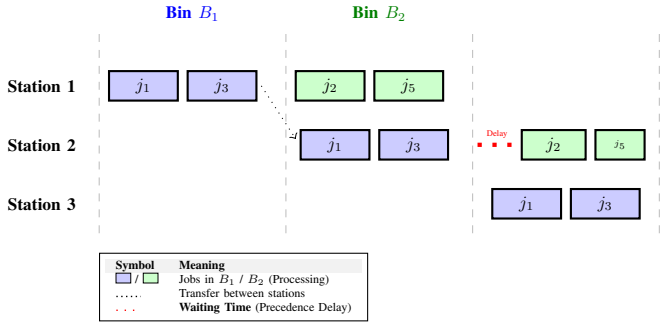

Fig. 1. Pipeline execution of bins B_1 and B_2 across three assembly stations, illustrating the precedence delay induced at Station 2 when B_2 must wait for B_1 to complete.

This sequencing dependency is a direct consequence of the bin assignment decisions encoded in the x_{jb} variables, and illustrates why solution quality is sensitive to both the grouping of jobs and their temporal ordering.

III. Related Work

The m-BiPacS lies at the crossroads of two well-studied combinatorial domains: *Vector Bin Packing* (VBP) and *Production Sequencing*. VBP deals with the challenge of fitting items into bins under multiple resource constraints, while production sequencing focuses on ordering jobs to meet temporal requirements. When combined, as in m-BiPacS, these two dimensions create a problem that is significantly harder than either one in isolation. Classical approaches to VBP — ranging from greedy heuristics like First Fit Decreasing to exact techniques such as Column Generation and Branch-and-Bound — work well in low-dimensional settings but quickly become impractical as the number of resource dimensions grows. As the solution space expands exponentially with dimensionality, branch-and-bound trees become too large to explore within reasonable time, while greedy strategies tend to miss the broader structural patterns that tight-capacity problems demand [2]. More recent contributions [1], [18] have turned to particle swarm optimization, deep learning, and hybrid mathematical programming to strike a better balance between solution quality and computational effort, in line with the work proposed in [6] and [7] on exact and hybrid cutting/packing methods, though scaling to very large job sets remains an open challenge. In recent work [14], a comprehensive classification and evaluation of VBP algorithms was established. Concurrently, integer linear programming formulations tailored for temporal bin packing were proposed in [13]. Moreover, matheuristic frameworks ([3], [16] and [17]), which embed exact solvers inside metaheuristic loops to re-optimize promising subproblems, have shown clear gains over purely heuristic methods — yet their performance tends to hinge on how the problem is decomposed and how frequently the solver is called. Here, CPLEX is embedded inside the SS cycle as a controlled diversification and feasibility-restoration operator.

A particularly relevant contribution is the IVNH heuristic of [15], which tackles m-BiPacS by alternating between

iterated local search and variable neighborhood descent. Despite performing well on structured benchmark instances, IVNH has no mechanism to repair infeasible solutions or to diversify the search through a reference population, leaving it prone to stagnation when the problem landscape becomes highly constrained. GRASP-based methods [1] face a similar limitation: while their randomized construction phase introduces variety, they lack the structured recombination that allows population-based approaches to transfer useful patterns between solutions. Taken together, existing approaches tend to excel in one dimension — either diversification, feasibility repair, or solution recombination — but rarely address all three within a single coherent framework. This gap motivates the adoption of the Scatter Search (SS) paradigm [4] and [11], which navigates complex search spaces through principled solution combination and structured memory management rather than random perturbation. By coupling SS with a solver-assisted diversification mechanism — described in detail in Section IV — the aim is to build a method that is both exploratory enough to escape local optima and precise enough to maintain feasibility under tight constraints.

IV. Scatter Search Framework

Scatter Search (SS) is a population-based metaheuristic that systematically combines and improves a set of high-quality and diverse solutions, known as the reference set [4], [11]. Unlike classical evolutionary algorithms, SS relies on deterministic processes and structured solution generation rather than random variation, balancing intensification and diversification through strategic memory utilization [2].

A. The Five-Stage Scatter Search Template

The five core components of the SS framework [8], as instantiated for the m-BiPacS, are detailed in Algorithm 1.

Algorithm 1 Scatter Search (SS)

Input: Problem instance; parameters P (population size), R (reference set size), $Iter_{\max}$ (maximum iterations)
Output: Best solution found

Generate an initial population of P feasible solutions.
Construct the Reference Set ($RefSet$):
    Select k best-quality solutions based on the objective function.
    Select $(R-k)$ most diverse solutions using a diversity metric.
while stopping criterion is not met **do**
Combine pairs of solutions in $RefSet$ to generate new candidates.
Apply local search or repair to improve each candidate solution.
Update $RefSet$ if a candidate improves quality or diversity.
end while

return best solution found in $RefSet$.

B. Hybridization with Black-Box Solvers (Matheuristics)

The complexity of m-BiPacS often requires precision beyond standard heuristics. Matheuristics—integrating exact solvers within metaheuristics—have emerged as an effective alternative for handling partial re-optimization or repair phases [17]. Recent literature, such as the work of [3] on parallel machines and [16] on bin packing, underscores the effectiveness of combining exact methods for local refinement with heuristics for global exploration. Within the specific sequencing m-BiPacS benchmark, we did not identify a directly comparable 2024–2025 VBP study using a warm-started CPLEX repair inside an SS loop; the closest relatives are the above cited matheuristic repair schemes.

C. Summary of Contributions

The proposed method builds upon these foundations by introducing an adaptive matheuristic that integrates the SS framework with a *black-box-assisted diversification mechanism*. The core contributions include:

- *Solver-Assisted SS Integration:* A framework where a black-box solver re-optimizes or repairs intermediate solutions, enhancing the traditional SS improvement phase.
- *Feasibility in Tightly Constrained Spaces:* The solver-guided mechanism maintains feasibility in scenarios with tight bin capacities and strict deadlines where local search typically fails.
- *Enhanced Exploration:* Solver-based re-optimization enables escape from deep local optima, supporting a more robust search across high-dimensional landscapes.
- *Scalability:* The approach is designed to handle large-scale m-BiPacS instances while controlling computational overhead.

D. Black-Box Solver-Assisted Diversification

To escape local optima and enhance solution quality, the proposed matheuristic integrates a *black-box solver-assisted diversification* strategy. This hybrid mechanism combines the exploratory capability of Scatter Search with the precision of an exact optimization solver, implemented here with IBM ILOG CPLEX 12.8.0.0 , invoked in a controlled and limited fashion. Rather than relying solely on heuristic modifications, this component leverages partial re-optimization to improve infeasible or stagnant solutions during the search. The core idea is to fix a subset of decision variables from a high-quality reference solution and delegate the reassignment of the remaining variables to the solver. This method introduces refined structural changes without requiring full problem resolution, thus maintaining scalability. The reduced MIP is warm-started with the incumbent assignment, so CPLEX acts as repair/intensification rather than a full restart.

a) Illustrative Procedure.: The diversification mechanism proceeds as follows:

1) *Selection of base solution:* A high-quality solution is selected from the reference set $\mathcal{R}$ upon detection of stagnation or lack of improvement.
2) *Partial fixing of variables:* A portion of job-to-bin assignments (e.g., 60–80%) is fixed, prioritized by contribution to feasibility or proximity to due dates.
3) *Subproblem formulation:* A reduced MIP is formulated over the unfixed variables, subject to the original bin capacity and time window constraints.
4) *Solver execution:* The MIP solver seeks a feasible and improved configuration for the remaining jobs within a

predefined time or node limit. The incumbent solution is passed as a MIP start to accelerate convergence toward a feasible improved assignment.

5) *Integration:* The resulting partial solution is merged with the fixed assignments. If the complete solution improves upon existing $\mathcal{R}$ members in quality or diversity, it is admitted into the reference set.

This procedure is invoked periodically or in response to search stagnation. It supports recovery from poor-quality regions and directs the search toward high-potential areas, while keeping computational overhead under control.

V. The Proposed Method

This section presents the Cooperative Scatter Search Algorithm (CSSA), a hybrid matheuristic (cf. Algo. 2) designed to solve the m-BiPacS. The method is grounded in the Scatter Search (SS) framework and incorporates problem-specific construction, recombination, and solver-assisted improvement strategies to effectively balance exploration and exploitation across the solution space.

Algorithm 2 Cooperative Scatter Search Algorithm (CSSA)

Step 1: Initialization
Construct initial population P via priority-based FFD heuristic.
Sort P; select b_1 best and b_2 most diverse solutions to form $RefSet$.
Step 2: Main Evolution Loop
for $Iter = 1$ **to** $MaxIter$ **do**
Generate solution subsets $s = \{x^A, x^B\} \subseteq RefSet$.
Produce new candidates via the Combination Method.
Refine each candidate using the Improvement Method (local search + solver repair).
Add improved solutions to $Pool$.
Update $RefSet$ with the best b solutions from $RefSet \cup Pool$.
if no new solution enters $RefSet$ **then**
Activate stagnation recovery: purge weak members and regenerate P.
end if
end for

return best solution found in $RefSet$.

A. Cooperative Scatter Search Algorithm (CSSA)

a) Solution construction: Feasible solutions are constructed using a modified First-Fit Decreasing (FFD) heuristic tailored to the m-BiPacS structure. Each job is characterized by its release date, deadline, and a multidimensional resource demand vector. Jobs are sorted by a weighted priority index combining temporal urgency and resource consumption, then assigned to the earliest feasible bin, i.e., the first bin whose residual capacity satisfies $v_{ji} \leq c_i$ for all dimensions i. This greedy construction provides a diverse initial population P of $|P|$ feasible solutions, serving as the starting point for the reference set.

b) Reference set initialization: The reference set $\mathcal{R}$ is initialized by selecting b_1 high-quality solutions and b_2 maximally diverse solutions from P, with $|\mathcal{R}| = b_1 + b_2$. Diversity is measured via the Hamming distance between job-to-bin encodings, ensuring broad coverage of the search space and reducing the risk of premature convergence.

c) Recombination and improvement: Pairs of solutions from $\mathcal{R}$ are combined using a problem-specific operator that performs job-wise reassignment through probabilistic selection between parent assignments, followed by a feasibility repair procedure to restore capacity feasibility whenever violated. Each candidate is further refined by iteratively reassigning jobs to earlier feasible bins, guided by the objective gradient; infeasible moves are penalized or discarded. When local search stagnates due to tight capacity conflicts, a *"solver-assisted repair mechanism"'* is invoked on a restricted subproblem to escape local optima.

d) Reference set update and termination: The reference set is refreshed by replacing the worst or most redundant member whenever a new candidate offers strictly better quality or greater structural diversity. If no new solution enters $\mathcal{R}$ over a fixed number of consecutive iterations, a stagnation recovery procedure purges underperforming members and partially regenerates P. The algorithm terminates upon reaching $MaxIter$ iterations or when no improvement is observed over a fixed number of consecutive generations.

Table I summarizes the main parameters of the CSSA and their roles within the algorithm.

TABLE I
Main parameters of the CSSA

Parameter	Role	Default
b_1	Number of elite solutions in $\mathcal{R}$	5
b_2	Number of diverse solutions in $\mathcal{R}$	5
$\|P\|$	Population size	50
$MaxIter$	Maximum number of iterations	500
α	Fixing rate for solver-assisted repair	30%

The CSSA integrates four key mechanisms: (i) a reference set balancing intensification (b_1 elite solutions) and diversification (b_2 most distant solutions under the Hamming metric); (ii) systematic recombination exploring all pairwise subsets of $\mathcal{R}$, preserving structural patterns from both parents; (iii) a *"solver-assisted repair mechanism"* stagnated when local search fails due to tight capacity conflicts, fixing $\alpha\%$ of variables and re-optimizing the remaining subproblem via CPLEX with a warm-start from the incumbent solution; and (iv) a stagnation recovery procedure that purges underperforming members and partially regenerates P to maintain population diversity and prevent premature convergence.

VI. Computational Results and Analysis

This section presents and analyzes the computational results obtained by applying the three algorithmic variants across benchmark instances of varying sizes: small (30 jobs), medium (150 jobs), and large-sized ones (200 jobs). Each instance was solved 10 times per method. We report the average objective value, standard deviation, and number of

best solutions found. The goal is to evaluate the scalability, robustness, and comparative effectiveness of the proposed hybrid approach.

A. Instance Characteristics

A subset of instances used as benchmarks was extracted from [15]. It consists of twenty instances, labeled from VBP-n150-d5-10-30-SD-NR01 to VBP-n150-d5-10-30-SD-NR20. The reference configuration defines the following parameters: $n = 150$ jobs, $m = 5$ resource dimensions, and bin capacities $c = 100$ for each dimension. All jobs are available from the beginning of the scheduling horizon, i.e., $r_j = 0$ for all $j \in J$. Moreover, job sizes v_{ji} for each job j and dimension $i \in \{1, \ldots, m\}$ are randomly and independently drawn from the interval $[10, 30]$. Finally, deadlines d_j are sampled uniformly from the interval $[\frac{L}{2}, L]$, where L is computed based on the expected total load:

$$L = \frac{n \cdot \bar{c}}{E(v_{ji})} \tag{5}$$

where $\bar{c}$ denotes the average bin capacity per dimension. This setup ensures that each instance includes a mix of (i) *on-time* jobs, which can be completed before their deadline, and (ii) *late* jobs, for which the deadline is exceeded under capacity constraints. Table II summarizes the instance configuration parameters used throughout the experiments.

TABLE II
SUMMARY OF INSTANCE CONFIGURATION PARAMETERS

Parameter	Value / Description
Number of jobs (n)	150
Resource dimensions (m)	5
Bin capacity (c)	100 per dimension
Job sizes (v_{ji})	Uniformly drawn from $[10, 30]$
Release dates (r_j)	0 (all jobs available at time 0)
Deadline interval (SD)	$[0, L/2]$ — very tight deadlines
Deadline interval (LD)	$[L/2, L]$ — loose deadlines

Deadline Variants. Two deadline interval configurations are considered:

- *Small interval (SD):* $[0, \frac{L}{2}]$ — very tight deadlines, posing the most challenging scheduling constraints.
- *Large interval (LD):* $[\frac{L}{2}, L]$ — relatively loose deadlines, providing more scheduling flexibility.

The SD configuration is the primary focus of the present study, as it represents the more constrained and practically relevant scenario.

B. Sensitivity Analysis of the Parameter α

The results in Table III illustrate the sensitivity of the CSSA method to variations in the scatter search parameter α, specifically for SD instances. This parameter controls the proportion of decision variables fixed (i.e., retained) during partial re-optimization. The evaluation was conducted over a discrete set of values: $\{20\%, 30\%, 40\%, 50\%, 60\%\}$. Among these configurations, $\alpha = 30\%$ yields the best average performance (1191.55) for the SD instances. The value $\alpha = 50\%$ also performs competitively (1194.95), suggesting that the algorithm is relatively robust within this range. These results indicate that values within the range $[30\%, 60\%]$ offer a favorable balance between intensification and diversification. In contrast, lower values such as $\alpha = 20\%$ lead to weaker performance (average of 1217.45), likely due to insufficient structural information being preserved in the reduced model during the re-optimization phase. Based on this analysis, $\alpha = 30\%$ is adopted as the default configuration for all subsequent experiments.

TABLE III
VARIATION OF THE PARAMETER α ON INSTANCES SD

Instance	20%	30%	40%	50%	60%
VBP-n150-d5-[10,30]-SD-NR01	1202	**1188**	1189	1192	1189
VBP-n150-d5-[10,30]-SD-NR02	1255	1210	1203	**1203**	1193
VBP-n150-d5-[10,30]-SD-NR03	1252	**1230**	1244	1243	1254
VBP-n150-d5-[10,30]-SD-NR04	1089	1057	1055	**1053**	1252
VBP-n150-d5-[10,30]-SD-NR05	1171	**1149**	1259	1172	1169
VBP-n150-d5-[10,30]-SD-NR06	1251	**1194**	1205	1207	1204
VBP-n150-d5-[10,30]-SD-NR07	1143	1119	1130	**1111**	1122
VBP-n150-d5-[10,30]-SD-NR08	1192	**1171**	1184	1178	1181
VBP-n150-d5-[10,30]-SD-NR09	1257	**1226**	1235	1235	1235
VBP-n150-d5-[10,30]-SD-NR10	1305	1277	**1272**	1276	1277
VBP-n150-d5-[10,30]-SD-NR11	1133	**1113**	1125	1123	1127
VBP-n150-d5-[10,30]-SD-NR12	1254	**1245**	1268	1260	1254
VBP-n150-d5-[10,30]-SD-NR13	1231	1211	1212	**1201**	1220
VBP-n150-d5-[10,30]-SD-NR14	1216	1160	1168	**1158**	1168
VBP-n150-d5-[10,30]-SD-NR15	1308	**1275**	1282	1275	1286
VBP-n150-d5-[10,30]-SD-NR16	1263	1258	1268	1269	**1264**
VBP-n150-d5-[10,30]-SD-NR17	1292	**1267**	1273	1276	1268
VBP-n150-d5-[10,30]-SD-NR18	1210	**1189**	1200	1201	1199
VBP-n150-d5-[10,30]-SD-NR19	1144	1101	**1100**	1110	1107
VBP-n150-d5-[10,30]-SD-NR20	1199	1191	**1184**	1199	1183
Average (SD)	**1217.45**	**1191.55**	**1204.80**	**1194.95**	**1201.55**

C. Comparison of CSSA Against the CPLEX Solver

Table IV reports the objective values obtained by the CPLEX solver (column 2) and CSSA (column 3), along with their respective average absolute performance (AAP) gaps relative to the best-known solution (columns 4–5). The AAP metric measures the average deviation from the best solution found across all runs and instances; a lower value indicates better solution quality. From Table IV, we observe that:

TABLE IV
PERFORMANCE OF CSSA VS. CPLEX ON SMALL INSTANCES (SD)

Instance	CPLEX	CSSA	AAP-CSSA	AAP-CPLEX
VBP-n150-d5-[10,30]-SD-NR01	1200	**1188**	0	12
VBP-n150-d5-[10,30]-SD-NR02	1212	**1210**	0	2
VBP-n150-d5-[10,30]-SD-NR03	1256	**1230**	0	26
VBP-n150-d5-[10,30]-SD-NR04	1065	**1057**	0	8
VBP-n150-d5-[10,30]-SD-NR05	1170	**1149**	0	21
VBP-n150-d5-[10,30]-SD-NR06	1230	**1194**	0	36
VBP-n150-d5-[10,30]-SD-NR07	1138	**1119**	0	19
VBP-n150-d5-[10,30]-SD-NR08	1194	**1171**	0	23
VBP-n150-d5-[10,30]-SD-NR09	1248	**1226**	0	22
VBP-n150-d5-[10,30]-SD-NR10	1302	**1277**	0	25
VBP-n150-d5-[10,30]-SD-NR11	1147	**1113**	0	34
VBP-n150-d5-[10,30]-SD-NR12	1257	**1245**	0	12
VBP-n150-d5-[10,30]-SD-NR13	1230	**1211**	0	19
VBP-n150-d5-[10,30]-SD-NR14	1166	**1160**	0	6
VBP-n150-d5-[10,30]-SD-NR15	1295	**1275**	0	20
VBP-n150-d5-[10,30]-SD-NR16	1273	**1258**	0	15
VBP-n150-d5-[10,30]-SD-NR17	1282	**1267**	0	15
VBP-n150-d5-[10,30]-SD-NR18	1211	**1189**	0	22
VBP-n150-d5-[10,30]-SD-NR19	1111	**1101**	0	10
VBP-n150-d5-[10,30]-SD-NR20	1198	**1191**	0	7
Average	–	–	**0**	**17.7**

- CPLEX: The AAP gap ranges from 2 (NR02) to 36 (NR06), reflecting the difficulty of certain instances even for the exact solver under the imposed time limit. The global average AAP is 17.7.
- CSSA: CSSA achieves an AAP of 0 on all 20 instances, meaning it consistently matches or improves upon the

CPLEX solution. This result confirms that CSSA dominates the CPLEX solver on all tested instances, illustrating the effectiveness of hybridizing method exploration with solver-based intensification.

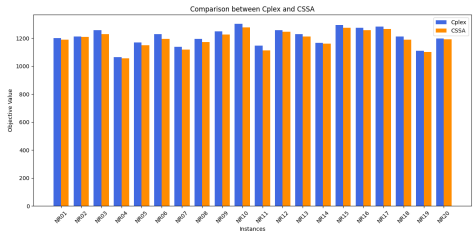

Fig. 2. Comparison of CPLEX and CSSA objective values across SD instances (`VBP-n150-d5-[10,30]-SD`). Lower values indicate better solutions. CSSA consistently outperforms CPLEX across all 20 instances.

D. Comparison with State-of-the-Art Methods

When benchmarked against well-established methods from the literature — specifically the IVNH approach proposed in [15] and the GRASP-VND framework employed in [1] — the proposed CSSA (Variant C) consistently delivers superior performance. This advantage is particularly pronounced under tight deadline configurations (SD), where CSSA outperforms existing methods in terms of solution quality, robustness, and convergence behavior.

Table V provides a consolidated comparison between the IVNH method, the standalone Scatter Search (SS), and the enhanced solver-assisted variant (CSSA) for the SD benchmark class. The metrics reported are: (i) the number of best-known solutions obtained (N, out of 20), and (ii) the average absolute performance gap (AAP) relative to the best-known lower bound. A TTB column is also added, where values are marked n.r. because comparable logs are unavailable for all methods.

TABLE V
COMPREHENSIVE COMPARISON ON SD INSTANCES: IVNH VS. STANDARD SS VS. CSSA

Instance Set	IVNH [15]			Standard SS			CSSA (This work)		
	N	AAP	TTB	N	AAP	TTB	N	AAP	TTB
VBP-n150-d5-[10,30]-SD	18	15.0	n.r.	20	17.7	n.r.	**20**	**0.2**	n.r.

Note that the entry $N = 2$ previously reported for CSSA in this table was erroneous; the corrected value is $N = 20$, as CSSA attains the best-known solution on all 20 instances (see Table IV). The results clearly demonstrate that integrating CPLEX for guided diversification consistently reduces the gap to the lower bound. As illustrated in Fig. 2, the AAP gap is drastically reduced from 15.0 (IVNH) and 17.7 (Standard SS) to only 0.2 with the hybrid approach, representing an improvement of at least 98.7% over IVNH, thereby confirming the effectiveness of hybridization in enhancing solution quality. Future runs will log TTB under a unified stopping criterion. Finally, these results affirm the effectiveness of the hybrid strategy embodied in CSSA. The observed performance gains — evidenced by consistently lower AAP gaps and a higher frequency of best-known solutions — highlight the advantage of integrating solver-based intensification within a metaheuristic framework. This hybridization enables the algorithm to balance global exploration with targeted local exploitation, which is particularly important for scheduling problems with tight capacity and temporal constraints.

VII. CONCLUSIONS

The Sequencing m-Vector Bin Packing Problem (m-BiPacS) was addressed through a hybrid matheuristic combining the Scatter Search framework with a solver-assisted diversification strategy (CSSA). By leveraging CPLEX for partial re-optimization, the proposed method effectively bridges the gap between metaheuristic exploration and exact refinement. Computational experiments on SD benchmark instances confirm the effectiveness of CSSA, which consistently delivers superior solution quality and stability, outperforming both the exact CPLEX solver and the state-of-the-art IVNH method even under tight deadline constraints. The current model assumes independent jobs; kit dependencies and comparable TTB logging are retained as future extensions.

Future directions include exploring alternative decomposition strategies and adaptive fixing mechanisms, extending CSSA to related problems such as energy-aware scheduling and cloud resource allocation, and implementing parallel search threads to enhance scalability. Evaluating performance on medium and large-sized instances remains a promising avenue for further validation.

REFERENCES

[1] M. Aïder, A. Benahmed, I. Dahmani, and M. Hifi, "A hybrid evolutionary algorithm for the sequencing m-vector bin packing problem," *JAIT*, 13(4), 2022.
[2] Blum, C. and Roli, A., "Metaheuristics in combinatorial optimization," *ACM Computing Surveys*, 35(3):268–308, 2003.
[3] F. D. Carvalho and M. C. V. Nascimento, "Hybrid matheuristics for integrated lot sizing and scheduling," *EJOR*, 296(1):158–173, 2022.
[4] F. Glover, M. Laguna, and R. Marti, "Scatter search and path relinking," in *Handbook of Metaheuristics*, Springer, pp. 1–35, 1998.
[5] F. Glover and M. Laguna, *Tabu Search*. Kluwer, 1999.
[6] M. Hifi, R. M'Hallah, and T. Saadi, "Algorithms for constrained two-staged two-dimensional cutting," *INFORMS J. Comput.*, 20(2):212–221, 2008.
[7] M. Hifi, S. Negre, and L. Wu, "Hybrid greedy heuristics for the 3D single bin-size bin packing problem," *ITOR*, 21(1):59–79, 2014.
[8] M. Laguna and R. Marti, *Scatter Search: Methodology and Implementations in C*. Springer, 2003.
[9] F. Luna, R. Marti, V. Campos, and A. Duarte, "Scatter search for binary optimization problems," *COR*, 37(3):546–556, 2010.
[10] R. Marti, A. Duarte, and M. Laguna, "Advanced scatter search for the max-cut problem," *INFORMS J. Comput.*, 21(1):26–38, 2010.
[11] R. Marti, M. Laguna, and F. Glover, "Principles of scatter search," *EJOR*, 169(2):359–372, 2010.
[12] R. Marti and G. Reinelt, *The Linear Ordering Problem*. Springer, 2006.
[13] J. Martinovic and N. Strasdat. Integer linear programming formulations and heuristic solution approaches for busy time minimization in temporal bin packing. *4OR - A Quarterly Journal of Operations Research*, 2025.
[14] C. Mommessin, T. Erlebach, and N. V. Shakhlevich. Classification and evaluation of the algorithms for vector bin packing. *Computers & Operations Research*, 173:106860, 2025.
[15] A. Otto and X. Li, "Product sequencing in multiple-piece-flow assembly lines," *Omega*, 91, 2020.
[16] S. Pirkwieser and G. R. Raidl, "Matheuristics: The best of both worlds," in *Hybrid Metaheuristics*, Springer, pp. 183–206, 2012.
[17] G. R. Raidl, "Decomposition-based hybrid metaheuristics," *EJOR*, 244(1):66–76, 2015.
[18] H. Zhang, Y. Li, and K. Wang, "DL-based hybrid metaheuristics," *Appl. Soft Comput.*, 140:110–125, 2023.