A Fraction of Pareto Nodes in DDs

Table 4: Percentage (rounded) of Pareto nodes across different sizes (number of objectives, number of variables) for 10 instances per size.

MOSPP	Pareto Node (%)	MOKP	Pareto Node (%)	MOTSP	Pareto Node (%)
(3, 100)	1	(3, 80)	3	(3, 15)	2
(5, 100)	1	(5, 40)	7	(4, 15)	6
(7, 100)	2	(7, 40)	12		

Table 4 reports the percentage of Pareto nodes observed across three benchmark problem classes—MOSPP, MOKP, and MOTSP—for varying numbers of objectives and decision variables, averaged over ten instances per configuration. Overall, the results indicate that Pareto nodes constitute only a small fraction of the total nodes in a DD.

B Mathematical Notation

The notation used to describe MOILP and DDs is presented in Table 5 and Table 6, respectively.

Table 5: Mathematical notation for MOILP

Symbol	Description
$\mathcal{P}$	A multiobjective integer linear programming problem
N	Number of decision variables
K	Number of objectives
$x \in \mathbb{Z}^N$	Integer-valued decision vector
$\mathcal{X} \subseteq \mathbb{Z}^N$	Feasible solution set
$C \in \mathbb{R}^{K \times N}$	Objective function coefficient matrix
$\mathcal{Z}$	Set of feasible objective vectors (Image of ($\mathcal{X}$) in the objective space)
$\prec$	Dominance relation: $z^1 \prec z^2$ implies z^1 dominates z^2
$\mathcal{Z}^\star \subseteq \mathcal{Z}$	Pareto frontier (set of nondominated objective vectors)

C Problem Definition and Instance Generation

In this section, we provide the formulation of the MOILP problems considered in this work and corresponding instance generation scheme.

C.1 MOTSP

Problem Definition The MOTSP extends the classical traveling salesperson problem by incorporating multiple, potentially conflicting cost metrics associated

Table 6: Mathematical notation for DDs

Symbol	Description
$\mathcal{B} = (\mathcal{N}, \mathcal{A})$	A decision diagram for problem $\mathcal{P}$
$\mathcal{N}$	Set of nodes in a DD
$\mathcal{A}$	Set of arcs (directed edges) in a DD
$\mathcal{L}_j$	Set of nodes in layer j, for $j \in \{1, \ldots, N+1\}$
$\mathbf{r}, \mathbf{t}$	Root node (layer 1) and terminal node (layer $N+1$)
$\omega(\mathcal{B})$	Width of the DD: $\max_j \lvert\mathcal{L}_j\rvert$
$a \in \mathcal{A}$	An arc in the DD
$d(a)$	Assignment value of arc a (value for variable x_j)
$v(a)$	Objective weight of arc a (vector in $\mathbb{R}^K$)
p	A path from $\mathbf{r}$ to $\mathbf{t}$
$x(p)$	Solution vector encoded by path p
$v(p)$	Objective vector accumulated along path p
$\mathcal{X}_\mathcal{B}$	Set of feasible integer solutions encoded by $\mathcal{B}$
$\mathcal{Z}_\mathcal{B}$	Set of objective vectors encoded by $\mathcal{B}$
$\mathcal{Z}^\star_\mathcal{B}$	Set of nondominated vectors in $\mathcal{Z}_\mathcal{B}$ (Pareto frontier)
Restricted DDs and Construction	
$\mathcal{B}'$	Restricted DD where $\mathcal{X}_{\mathcal{B}'} \subseteq \mathcal{X}$
W	Maximum width parameter for restricted DDs
$\mathcal{Z}^\star_{\mathcal{B}'}$	Approximate Pareto frontier derived from $\mathcal{B}'$
$\mathcal{S}$	State space defined by the DP formulation of $\mathcal{P}$
$s(u) \in \mathcal{S}$	State associated with node u

with traversing arcs. We consider a complete directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1, \ldots, N\}$ is the set of vertices (cities) and $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$ is the set of edges. There are K cost matrices, where c^k_{ij} denotes the cost of edge (i, j) for the k-th objective. The problem consists of finding a Hamiltonian cycle (a permutation of the vertices) that simultaneously minimizes the objective vectors. In the context of the MOILP formulation, the decision variables $x = (x_1, \ldots, x_N) \in \mathcal{V}^N$ represent the permutation of vertices visited, such that x_j is the index of the vertex visited at step j.

Instance Generation The instances are generated as in [28]. For each K, we generated integer coordinates for N cities on a 1000×1000 square (uniformly at random) and used Euclidean distances to create the distance matrix.

C.2 MOKP

Problem Definition The MOKP involves selecting a subset of items to maximize multiple profit objectives subject to a single weight capacity constraint. Given N items, let $w \in \mathbb{Z}^N_+$ be the vector of item weights and $B \in \mathbb{Z}_+$ be the knapsack capacity. The profit of item j for the k-th objective is given by the entry C_{kj}. The problem is formulated as:

$$\max_{x \in \{0,1\}^N} \left\{ Cx \mid \sum_{j=1}^{N} w_j x_j \leq B \right\}.$$

Here, the decision variable $x_j = 1$ implies item j is selected, and $x_j = 0$ otherwise.

Instance Generation We follow the instance generation scheme used in [22], where each profit C_{kj} and weight w_j were drawn uniformly at random from the integer interval $[1, \ldots, 1000]$. The capacity of the knapsack was set to $B := \lceil 0.5 \sum_{j=1}^{N} w_j \rceil$.

C.3 MOSPP

Problem Definition The MOSPP seeks to select a subset of variables to maximize objectives subject to pairwise conflict constraints (or packing constraints). Let $A \in \{0,1\}^{M \times N}$ be a binary constraint matrix with M constraints, and $\mathbf{1}$ be a vector of ones of size M. The problem is defined as:

$$\max_{x \in \{0,1\}^N} \{Cx : Ax \leq \mathbf{1}\}.$$

The constraint $Ax \leq \mathbf{1}$ implies that for any row m of A, at most one variable j with $A_{mj} = 1$ can be set to 1. This is equivalent to finding a maximum weight independent set on the intersection graph defined by A.

Instance Generation The instance generation for the MOSPP is based on the previous work by [33]. Specifically, we fix the number of constraints $M = N/5$. Then for each constraint, we select the number of variables that participate in it from a random uniform distribution over [2, 20], resulting in an average of 10 variables per constraint.

We identified a minor issue in this instance generation process. Specifically, certain variables were not involved in any constraints. To address this, we randomly assign such variables to any one of the existing constraints, ensuring all variables participate meaningfully in the problem formulation.

D Implementation Details of Node Selection Heuristics

In this section, we describe the implementation of the node selection heuristics for the MOKP and MOSPP. The corresponding details for the MOTSP are presented in Section 3.3.

D.1 Multiobjective knapsack problem

State Definition The state $s(u)$ at layer j (where the decision for item $j-1$ has just been made) tracks the resource consumption. It is defined as a scalar:

$$s(u) = q, \tag{6}$$

where $q \in \mathbb{Z}_{\geq 0}$ represents the accumulated weight of items selected in the path from the root to node u. Formally, if u is reached by a path p with assignments $x_1, \ldots, x_{j-1}$, then $s(u) = \sum_{t=1}^{j-1} w_t x_t$. A transition $x_j = 1$ is feasible only if $s(u) + w_j \leq B$. If $x_j = 1$, the next state is $s(u) + w_j$; if $x_j = 0$, the next state is $s(u)$.

Rule-based Heuristic As the state in MOKP is scalar-valued, the NOSH-Rule method uses Scal+ to select the nodes along with minWeight variable ordering [32]. In minWeight variable ordering, we sort the items based on the ascending order of their weight before constructing the DD.

Learning-based Heuristic The features used by the NOSH-ML method are presented in Table 7. Note that these features rely on the fact that the problem has only one constraint. Extending this approach to problems with multiple constraints can be non-trivial.

Table 7: Features associated with a DD node for the MOKP.

Feature scope	**Feature**	**Count**
Instance features	The number of objectives	1
	The number of items (or variables)	1
	The capacity of the knapsack	1
	The mean, min., max., and std. of the weights	4
	The mean, min., max., and std. of the values	12
Layer-variable features	The weight associated with variable	1
	Average value across objectives	1
	Maximum value across objectives	1
	Minimum value across objectives	1
	Standard deviation across objectives	1
	Ratio of average value across objectives to weight	1
	Ratio of maximum value across objectives to weight	1
	Ratio of minimum value across objectives to weight	1
Layer-index features	Normalized layer index	1
State features	Normalized state (weight of the knapsack at the current node)	1
	Ratio of state by capacity	1
Total		44

D.2 Multiobjective set packing problem

State Definition The state $s(u)$ at layer j must capture the availability of the remaining variables $x_j, \ldots, x_N$ given the decisions made so far. The state is defined as the set of *eligible* variables:

$$s(u) = \mathcal{V}_{eligible} \subseteq \{j, \ldots, N\}, \tag{7}$$

where $k \in \mathcal{V}_{eligible}$ implies that selecting variable k (setting $x_k = 1$) does not violate any constraints given the variables already selected in the path to u. At the root node, $s(\mathbf{r}) = \{1, \ldots, N\}$. Given a node u at layer j with state $s(u)$:

– If $x_j = 0$, the constraint set remains unchanged for the remaining variables. The next state is $s(u) \setminus \{j\}$.

– If $x_j = 1$ (valid only if $j \in s(u)$), we must remove j and all future variables $k > j$ that conflict with j (i.e., variables k such that $\exists m, A_{mj} = 1 \wedge A_{mk} = 1$). The next state is $\{k \in s(u) \setminus \{j\} \mid \text{variable } k \text{ does not conflict with } j\}$.

Rule-based Heuristic The NOSH-Rule method uses the Card+ heuristic to select states as they are represented as a set. Ties among nodes with the same cardinality are broken by randomly selecting a subset of them to fit the width limit. To build the DD, we use the minState [6] variable ordering, where we select the variable appearing in the minimum number of states in the current layer to construct the next layer.

E Additional Computational Details

E.1 Width Selection for Restricted DDs

One of the key considerations in constructing restricted DDs is determining its width. If the width is too small, the DD may be overly restrictive, making it difficult to recover the true Pareto frontier. Conversely, if the width is too large, the performance of the restricted DD may closely resemble that of the exact DD, offering little computational advantage.

In the DD literature, the width is typically chosen based on empirical validation, tuned to the downstream task performance using the restricted DD. In this work, we begin by setting the initial width to approximately 20% of the average width of the exact DDs for a given problem class and instance size. If the method achieves both cardinality and precision above 80%, we retain the current width. Otherwise, we incrementally increase the width budget by 10%. Conversely, if the method performs exceptionally well with the initial width, we systematically reduce it to identify a minimal effective width.

As demonstrated in Section 5, effective width budgets for the MOSPP, MOKP, and MOTSP are found to be 1%, 30%, and 20% of the average exact DD width, respectively.

E.2 Hyperparameter Configuration

Appendix E.2 provides the architectural and training details for the learning-based heuristic for the MOTSP. Appendix E.2 outlines the configuration of the learning-based heuristic for the MOKP, which is based on XGBoost. Finally, Appendix E.2 describes the parameters of the NSGA-II-based evolutionary baseline.

MOTSP scorer configuration We enrich the raw vertex features of the MOTSP by computing additional features based on the distances per objective. Specifically, for each vertex, we compute the mean, minimum, maximum, standard deviation, and interquartile range (75th percentile minus 25th percentile) of the distances to other nodes for each objective. These five statistical features

Table 8: Hyperparameter configuration for the XGBoost-based scorers used in MOKP

	Size		
Parameter	(7, 40)	(4, 50)	(3, 80)
max_depth	9	7	5
min_child_weight	10000	10000	1000
eta	0.3		
objective	binary:logistic		
num_round	250		
early_stopping_rounds	20		
eval_metric	logloss		

are concatenated with the original 2D coordinates of the vertex, resulting in a 7-dimensional feature vector for each vertex. This enhanced representation provides the model with richer contextual information about each node's relative position and importance in the instance.

The Objective Aggregator processes this input using a Deep Sets architecture, which is permutation invariant over objectives. It transforms the enriched node features $h_0^v \in \mathbb{R}^{N\times K\times 7}$ and edge features $h_0^e \in \mathbb{R}^{N\times N\times K}$ into fixed-size embeddings $h_{\text{agg}}^v \in \mathbb{R}^{N\times 32}$ and $h_{\text{agg}}^e \in \mathbb{R}^{N\times N\times 32}$, respectively, with an embedding dimension of 32. These embeddings are then passed into the downstream Graph Transformer network.

Concretely, we define feature encoders and aggregators using functions:

$$\gamma_v : \mathbb{R}^7 \to \mathbb{R}^{32}, \quad \rho_v : \mathbb{R}^{64} \to \mathbb{R}^{32}, \quad \gamma_e : \mathbb{R} \to \mathbb{R}^{32}, \quad \rho_e : \mathbb{R}^{32} \to \mathbb{R}^{32}$$

Here, γ_v and γ_e are per-objective encoders applied independently to each node and edge for a given objective, while ρ_v and ρ_e are permutation-invariant aggregators that combine the representations across objectives. The output of this aggregation serves as a unified embedding that captures information across all objectives. These encoders and aggregators are implemented as a single linear layer with ReLU activation in the output.

The EGT is configured with 4 layers and 8 heads per layer and no dropout. The architecture uses ReLU as activation and omits biases in both multi-head attention and linear layers. A hidden-to-input dimension ratio of 2 is used in the MLP blocks. The model is trained for 100 epochs using the Adam optimizer. The learning rate is linearly warmed up from 5×10^{-5} to 5×10^{-4}, after which it follows a cosine decay schedule down to a minimum of 5×10^{-5}.

MOKP scorer configuration Table 8 describes the configuration details for the XGBoost-based node scorer for MOKP, including the number of trees, learning rate, and maximum depth.

Table 9: NSGA-II Configuration

Config	MOKP & MOSPP	MOTSP
Crossover	TwoPointCrossover	OrderCrossover
Mutation	BitflipMutation	InversionMutation
Sampling	BinaryRandomSampling	PermutationRandomSampling

Table 10: MOSPP results averaged over 100 test instances. Each column corresponds to a specific instance size (N, K), with rows giving the metrics for the Exact and `NOSH-Rule` methods. Refer to Section 4 for column description.

		$N = 100$					$N = 150$				
Metric	Method	$K = 3$	$K = 4$	$K = 5$	$K = 6$	$K = 7$	$K = 3$	$K = 4$	$K = 5$	$K = 6$	$K = 7$
Width	Exact	5,766	6,034	5,936	5,976	5,707	471,602	518,556	464,330	468,787	590,908
	`NOSH-Rule`	50	50	50	50	50	5,000	5,000	5,000	5,000	5,000
Time	Exact	0.045	0.198	1.610	4.496	30.167	10.030	50.061	260.660	566.015	782.735
	NSGA-II	1	1	1	5	23	1	10	83	334	381
	`NOSH-Rule`	0.002	0.034	0.602	1.553	12.212	0.415	6.910	76.564	182.013	313.181
Cardinality	Exact	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	NSGA-II	0	–	–	–	–	–	–	–	–	–
	`NOSH-Rule`	0.851	0.838	0.885	0.866	0.870	0.993	0.994	0.988	0.993	0.994
Precision	Exact	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	NSGA-II	0	–	–	–	–	–	–	–	–	–
	`NOSH-Rule`	0.892	0.899	0.942	0.941	0.931	0.995	0.996	0.992	0.999	0.999
$\lvert\hat{\mathcal{Z}}^\star\rvert$	Exact	238	1,117	4,765	9,117	25,457	787	6,099	29,061	59,951	103,489
	NSGA-II	4	–	–	–	–	–	–	–	–	–
	`NOSH-Rule`	226	1,051	4,591	8,550	24,534	786	6,089	28,953	59,724	103,275

NSGA II Configuration The crossover, mutation, and sampling strategies for the NSGA-II algorithm applied to different problems is provided in Table 9. For each problem type and size, we set the time budget for MOSPP, MOKP, and MOTSP to match the average runtime of `NOSH-Rule`, `NOSH-ML-FE`, and `NOSH-ML-E2E`, respectively. Initially, we set the population size to the average number of nondominated solutions observed for each problem size. However, this led to out-of-memory errors for some sizes, prompting us to reduce the population size accordingly.

F Additional results

Table 10, Table 11, and Table 12 present the complete NSGA-II results for MOSPP, MOKP, and MOTSP, respectively. In addition, Table 12 includes the performance of various rule-based NOSH variants explored in our experiments.

Table 11: MOKP results averaged across 100 test instances. Refer to Section 4 for column description.

N	K	Method	Width	Time	Cardinality	Precision	$\lvert\hat{\mathcal{Z}}^\star\rvert$	Inst.
40	7	Exact	9,709	147.431	1.000	1.000	25,098	92
		NSGA-II	–	900	0.060	–	–	99
		NOSH-Rule	2,000	0.879	0.188	0.643	4,131	92
			3,000	11.152	0.606	0.889	12,972	92
		NOSH-ML-FE	2,000	33.133	0.601	0.736	20,482	92
			3,000	60.814	0.881	0.956	22,267	92
50	4	Exact	12,359	25.995	1.000	1.000	3,564	100
		NSGA-II	–	120	0.010	–	–	100
		NOSH-Rule	2,500	0.955	0.170	0.360	1,132	100
			3,500	3.585	0.520	0.699	2,256	100
		NOSH-ML-FE	2,500	6.255	0.609	0.704	3,062	100
			3,500	11.631	0.875	0.920	3,367	100
80	3	Exact	20,097	127.627	1.000	1.000	2,442	100
		NSGA-II	–	300	0.340	–	–	100
		NOSH-Rule	4,000	9.198	0.096	0.187	1,015	100
			6,000	31.465	0.623	0.712	1,954	100
		NOSH-ML-FE	4,000	33.068	0.459	0.529	2,039	100
			6,000	57.097	0.925	0.948	2,363	100

Table 12: MOTSP results averaged across 100 test instances. Refer to Section 4 for column description. Note that methods prefixed with `Ord` correspond to different rule-based NOSHs.

N	K	Method	Width	Time	Cardinality	Precision	$\lvert\hat{\mathcal{Z}}^\star\rvert$	Inst.
15	3	Exact	24,024	4.843	1.000	1.000	868	100
		NSGA-II	-	3	0	0		100
		OrdMeanHigh	4,804	1.074	0.004	0.009	376	100
		OrdMeanLow	4,804	1.125	0.011	0.021	439	100
		OrdMaxHigh	4.804	1.085	0.003	0.006	366	100
		OrdMaxLow	4,804	1.114	0.013	0.027	425	100
		OrdMinHigh	4,804	1.076	0.003	0.007	369	100
		OrdMinLow	4,804	1.114	0.008	0.016	440	100
		NOSH-ML-E2E	4,804	1.356	0.912	0.947	832	100
	4	Exact	24,024	52.726	1.000	1.000	9,210	100
		NSGA-II	-	25	0	0		100
		OrdMeanHigh	4,804	1.733	0.004	0.015	2,737	100
		OrdMeanLow	4,804	2.632	0.014	0.035	3,254	100
		OrdMaxHigh	4,804	1.759	0.005	0.016	2,721	100
		OrdMaxLow	4,804	2.357	0.010	0.027	3,281	100
		OrdMinHigh	4,804	1.837	0.003	0.009	2,790	100
		OrdMinLow	4,804	2.360	0.008	0.020	3,229	100
		NOSH-ML-E2E	4,804	12.134	0.885	0.946	8,546	100