Graph Learning Assisted Multi-Objective Integer Programming

Yaoxin Wu[1], **Wen Song**[2], **Zhiguang Cao**[3,*], **Jie Zhang**[1],
Abhishek Gupta[3] **and Mingyan Simon Lin**[3]

[1]Nanyang Technological University
[2]Institute of Marine Science and Technology, Shandong University, China
[3]Singapore Institute of Manufacturing Technology, A*STAR
wuyaoxin@ntu.edu.sg, wensong@email.sdu.edu.cn, zhiguangcao@outlook.com
zhangj@ntu.edu.sg, {abhishek_gupta, simon_lin}@simtech.a-star.edu.sg

Abstract

Objective-space decomposition algorithms (ODAs) are widely studied for solving multi-objective integer programs. However, they often encounter difficulties in handling scalarized problems, which could cause infeasibility or repetitive nondominated points and thus induce redundant runtime. To mitigate the issue, we present a graph neural network (GNN) based method to learn the reduction rule in the ODA. We formulate the algorithmic procedure of generic ODAs as a Markov decision process, and parameterize the policy (reduction rule) with a novel two-stage GNN to fuse information from variables, constraints and especially objectives for better state representation. We train our model with imitation learning and deploy it on a state-of-the-art ODA. Results show that our method significantly improves the solving efficiency of the ODA. The learned policy generalizes fairly well to larger problems or more objectives, and the proposed GNN outperforms existing ones for integer programming in terms of test and generalization accuracy.

1 Introduction

Practical combinatorial optimization often involves multiple objectives that conflict with each other. Such problems can be termed as multi-objective combinatorial optimization (MOCO), which are widely studied in operations research and computer science [1], and also have broad applications in energy [2], engineering [3], biology [4], etc. In contrast to the single-objective optimization, MOCO centers on searching solutions with nondominated images (points) in the objective space, which constitute the Pareto front. These solutions reflect different preferences on the objectives, so that decision makers could select from them according to practical use. In this paper, we focus on multi-objective integer programming (MOIP), which provides a unified modeling framework for MOCOs, and is convenient in handling complex constraints which could be difficult for heuristic methods [5]. However, MOIP problems are extremely difficult to be solved optimally (i.e. enumerate the whole Pareto front) [6, 7]. Therefore, accelerating the computation of MOIPs to ameliorate the partial Pareto front in reasonable time is a critical and challenging issue [8, 9, 10].

Recently, deep learning has been extensively explored in solving single-objective combinatorial optimization problems such as vehicle routing, scheduling, and bin packing [11, 12, 13, 14, 15, 16, 17, 18]. As a general modeling and solving framework, single-objective integer programming (IP) has also received increasing attention, and this line of works mostly attempt to improve ingredients in the branch and bound (BnB) paradigm [19, 20, 21, 22, 23, 24]. In contrast, research on applying deep

*Corresponding Author.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

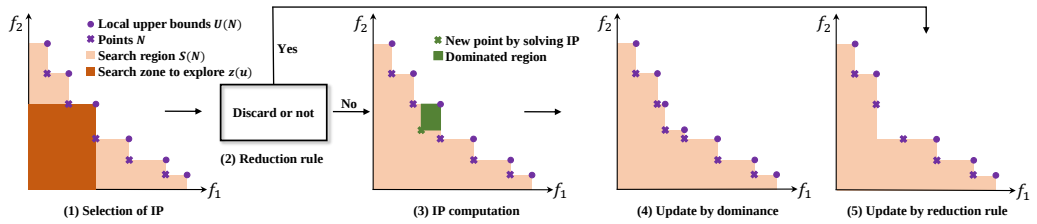

Figure 1: **An example of ODA for solving a MOIP with two objectives.** The algorithmic procedure iterates the following phases, 1) selecting one from all local upper bounds and defining the corresponding scalarized problem (IP); 2) determining whether the IP is to be solved, if Yes 3) solving the IP by calling a solver; 4) updating the search region according to the dominance of the newly derived image (point), if No 5) discarding the selected local upper bound without calling a solver.

learning to MOCO is relatively sparse, with only recent trials targeting at specific problems such as travelling salesman and knapsack problem [10, 25, 26, 27] rather than the more general MOIP.

This paper, for the first time, presents a graph learning based method to assist MOIP solving by improving the bounded-time performance of objective-space decomposition algorithms (ODAs). With highly optimized IP solvers such as CPLEX and Gurobi, ODAs iteratively decompose a MOIP into multiple scalarized problems (i.e. IPs) by applying local upper bounds in the objective space and solve one of the IPs to update the bounds, as illustrated in Figure 1. Since ODAs involve computation primarily in the low-dimensional objective space which could be fulfilled by calling mature IP solvers, this paradigm has attracted much research in the literature [8, 28, 29, 30]. However, existing ODAs often solve large amounts of IPs that 1) are *infeasible* or 2) lead to *repetition* of nondominated points, which result in unnecessary and redundant computation.

To tackle this issue, we propose to learn reduction rules to discard local upper bounds that induce futile IPs pertaining to infeasibility or repetition in iterations of ODAs, so as to reduce unnecessary computation eventually. To this end, we model the algorithmic procedure of ODAs as a Markov decision process based on the analysis of classic reduction rules in ODAs. To resolve the challenges in representing MOIP solving states, we propose a novel two-stage graph neural network (GNN) to parameterize the policy (reduction rule). In the first stage, the variable, constraint and objective nodes attend to each other locally to update their embeddings; while in the second stage, the embeddings are aggregated into three hypernodes for each pair of node types to realize a better global representation. We train the GNN based policy with imitation learning so that it learns to discard local upper bounds of futile IPs. We employ our method to improve a state-of-the-art ODA [29], and extensive results on benchmark problems indicate its efficacy in accelerating the search of nondominated points in bounded time and generalizing to larger problems or more objectives. We also compare the proposed GNN with existing ones for IPs to verify its superiority in learning representations for MOIPs.

2 Related work

2.1 Objective-space decomposition for MOIP

A variety of methods have been proposed for MOIPs. Among them, ODAs attract much attention since they usually resort to highly-optimized IP solvers for solving the decomposed scalarized problems, with additional light computation only processing local bounds in a low-dimensional objective space. IP solutions to scalarized problems update local bounds to delimit a search region where nondominated points exactly exist. As representatives, Kirlik and Sayın [31] equip ϵ-constraint with p-1 dimensional boxes representing the search region, and iteratively split them to filter out the ones without nondominated points. Klamroth et al. [28] formally describe the search region with full dimensional zones whose amount is significantly less than that of boxes in [31], inducing less subroutines for IPs. Dächert et al. [32] accelerate the update of the search region with a specific neighborhood relation between local upper bounds. Boland et al. [33] define the search region by known efficient solutions and extra disjunctive constraints, which increased the computational complexity in IP solving. Tamby and Vanderpooten [29] point out the infeasibility issue and propose a state-of-the-art framework for MOIPs where infeasible programs could be avoided. However, this

method needs to restore all solved IPs and look up their feasible domains to check feasibility of the current IP in each iteration. Moreover, existing ODAs generally lack a capacity to detect repetitive IPs. This paper provides a graph learning based method to fast bypass infeasibility and repetition of nondominated points. Other less related works pertaining to ODAs can be found in [34, 35, 30].

2.2 Graph learning for integer programs

Graph neural networks (GNNs) have been explored to learn representations for single-objective IPs, which identify problem structures better than trivial neural networks (e.g. multilayer perceptron (MLP)) and enable generalization to different scales. Gasse et al. [21] describe IPs at each node in BnB search as bipartite graphs and use graph convolutional network (GCN) to learn branching rules. It inspires a number of subsequent works to employ miscellaneous GNNs for tackling IPs. Ding et al. [36] propose tripartite graph embedding for IPs to predict values of binary variables. Gupta et al. [37] consider a hybrid use of GCN and MLP to accelerate the branching in BnB. Nair et al. [24] adapt the bipartite GCN in [21] to learn both branching rules and a specific diving heuristic at the root node. Wu et al. [38] apply it to select variables in IPs for destroy operations in a large neighborhood search framework. Recently, Khalil et al. [39] evolve the bipartite GCN by substituting its message passing structure into GIN and GraphSAGE [40, 41], to separately train node selection rules and predict warm starts for BnB search. The existing graph learning based methods might be inadequate to represent MOIPs where multiple objective nodes need to be identified and the potential scalarized problems need to be discriminated. In this paper, we propose an effective graph encoding for MOIPs, and design a novel two-stage GNN to learn the corresponding graph representations. We update node embeddings by attention in objective-variable, variable-constraint graphs, and aggregate these (locally) advanced node embeddings into hypernodes to further gain a favorable graph representation.

3 Preliminaries

3.1 Multi-objective integer programs

The integer-constrained optimization with p objective functions are formally described as follows:

$$\begin{aligned} \min_{x} \quad & f(x) = (f_1(x), \dots, f_p(x)) \\ \text{s.t.} \quad & Ax \leqslant b, \;\; x \in \mathbb{Z}^{n_z} \times \mathbb{R}^{n-n_z}, \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $m, n, n_z \in \mathbb{N}$ and $n \geqslant n_z > 0$. The MOIP is specified from the above formulation with $n = n_z$. Given a feasible solution x, its image (point) in the objective space is $y = f(x) = (f_1(x), \dots, f_p(x))$, resulting in the set of feasible solutions $\mathcal{X}$ and their images $\mathcal{Y}$. In $\mathcal{Y}$, an image y is *weakly dominated* by y' (i.e. $y' \leqq y$), if $y'_i \leqslant y_i, \forall i \in \{1, \dots, p\}$; y is *dominated* by y' (i.e. $y' \leq y$), if $y' \leqq y$ and $y \neq y'$; y is *strictly dominated* by y' (i.e. $y' < y$), if $y'_i < y_i, \forall i \in \{1, \dots, p\}$. Accordingly, a solution $x \in \mathcal{X}$ is *efficient* or *weakly efficient* if its image $f(x)$ is not dominated or strictly dominated by any other image in the objective space. The common goal of MOIPs is searching the Pareto front that is the set of nondominated images $\mathcal{Y}_{nd} = \{f(x) | x \in \mathcal{X}_e\}$ derived from efficient solutions $\mathcal{X}_e$.

3.2 Objective-space decomposition algorithms

Following [28, 32, 29], we describe the search region by local upper bounds to locate the remaining nondominated points at each iteration in ODAs.

Definition 1. Given a set of points (images) N in p-dimensional objective space, the corresponding search region is defined as $S(N) = \{y \in \mathbb{R}^p | y' \nleqq y, \forall y' \in N; y^I \leqq y\}$, where y^I is the ideal point with $y^I_i = \min_{x\in\mathcal{X}} f_i(x) = \min_{y\in\mathcal{Y}_{nd}} y_i$.

To concisely represent the search region and facilitate its decomposition into scalarized problems, the (maximal) local upper bounds are introduced [28].

Definition 2. A set of points $U(N)$ is called (maximal) local upper bounds of N if and only if 1) $S(N) = \bigcup_{u\in U(N)} z(u)$, where $z(u) = \{y \in \mathbb{R}^p | y^I \leqq y < u\}$ is the search zone under local upper bound u; 2) $\forall u, u' \in U(N), u \nleqq u'$.

The "maximal" emphasizes the condition 2), i.e., $\forall u, u', z(u) \nsubseteq z(u')$. Given the current set of local upper bounds and a new nondominated point, the search region (with the local upper bounds) can be efficiently updated by excluding the dominated region of the point via the algorithm in [28]. Such update procedure keeps shrinking the search region to locate the remaining nondominated points, and we elaborate it in Appendix A.1.

Proposition 1. Given $u \in U(N)$ and a strictly monotone function[2] $\mathcal{F} : \mathcal{Y} \to \mathbb{R}$, the optimal solution to the following scalarized problem (IP) will derive a nondominated point in $\mathcal{Y}_{nd}$,

$$\Gamma(u) := \min\{\mathcal{F}(y) = \mathcal{F}(y_1, \ldots, y_p) | y \in \mathcal{Y}; y < u\}. \tag{2}$$

The common instantiations of $\mathcal{F}$ include ϵ-constraint, Lexicographic and weighted Tchebycheff methods [42, 8], which are detailed in Appendix A.2. Given the above concepts, the generic algorithmic procedure for ODAs starts from a (local) upper bound $U(\emptyset) = \{(M, \ldots, M)\}$ with M being a large value,[3] and comprises four components, 1) selection rule that picks u from $U(N)$ in Eq. (2); 2) reduction rule that detects feasibility of IP in Eq. (2); 3) IP solving with a solver (e.g. Gurobi, CPLEX); 4) update of search region $U(N)$ according to dominance or reduction rule. This algorithmic paradigm has the property in Proposition 2. We prove Proposition 1, 2 in Appendix A.3.

Proposition 2. ODAs generate the whole Pareto front for MOIP with finite iterations.

4 Methodology

In this section, we identify the critical issue about the infeasibility and repetition of nondominated points in ODAs, and propose to circumvent the solving of the corresponding IPs with a reduction rule learned automatically by GNN. We encode the generic algorithmic procedure of ODAs as a MDP, where the reduction rule is parameterized by a two-stage GNN and trained with imitation learning.

4.1 Problem statement: infeasibility and repetition reduction

In ODAs, most of the computation time in each iteration is consumed by IP solving (normally with mature solvers). However, existing algorithms are less effective in identifying and discarding local upper bounds that induce futile IPs with infeasible or repetitive nondominated points. The commonly used reduction rules rely on the inclusion relationship of solution spaces between IPs solved and to be solved [31, 33, 29]. We summarize this class of reduction rules as follows.

Observation 1. Given a scalarized problem $\Gamma(u)$, it is infeasible if there exists a $\Gamma(u')$ with, 1) the same $\mathcal{F}$ as in $\Gamma(u)$, and 2) $u \leqq u'$, which has been solved and returns null.

Since all solutions in $\Gamma(u)$ are subset of those in $\Gamma(u')$, they have been proved infeasible under the same objective function by $\Gamma(u')$. However, reduction rules based on observation 1 require $\Gamma(u')$ to be solved to prove infeasibilty, which induces extra runtime. We extend it to the optimality cases.

Proposition 3. Given a scalarized problem $\Gamma(u)$, it is infeasible if there exists a solved $\Gamma(u')$ with 1) the same $\mathcal{F}$ as in $\Gamma(u)$, 2) the optimal point y^* satisfying $\mathcal{F}(u) \leqslant \mathcal{F}(y^*)$, and 3) $u \leqq u'$.

Proof. Given $u \leqq u'$, the solution space of $\Gamma(u')$ contains the one of $\Gamma(u)$. Since the optimal point y^* has achieved the lower bound in $\Gamma(u')$, $\Gamma(u)$ cannot attain any feasible solution y' with $\mathcal{F}(y') < \mathcal{F}(u) \leqslant \mathcal{F}(y^*)$ based on the strictly monotone property in Proposition 1. If so, it contradicts with the optimality of y^* in $\Gamma(u')$.

While the instantiation for Observation 1 as the reduction rule can be found in [31, 33], Proposition 3 is used as a reduction rule with Lexicographic scalarization in [29]. However, the above methods need clearly more memory and extra runtime to trace or iterate over the IPs to check infeasibility. Meanwhile, the repetition of nondominated points also broadly exists in ODAs, where existing methods generally detect them by calling solvers, inducing extra runtime as well [31, 29, 30].

To curtail unnecessary solver calls that cause infeasibility and repetition, we incline to fast predict these status immediately after a local upper bound is selected with its IP formulation. In the following, we convert the algorithmic procedure of ODAs into a MDP, and learn a reduction rule by GNN to discard the local upper bounds that may induce infeasible and repetitive points in corresponding IPs.

[2] A strictly monotone function ensures that: $\forall y, y' \in \mathbb{R}^p$, if $y \leq y'$, then $\mathcal{F}(y) < \mathcal{F}(y')$.

[3] In practice, M could be provided by a decision maker to constrain the search region of interest [28].

Remark. Since we learn the reduction rule to accelerate the detection of unnecessary IPs, an undesirable byproduct is that some IPs with effective nondominated points could be ignored, which may impair the completeness of the found Pareto points. However, our learned reduction rule has favorable potential to achieve a good trade-off that it is able to search reasonably more nondominated points in the bounded time, especially for large-scale problems.

4.2 Markov decision process

For each MOIP instance, we describe the solving process of ODAs as a MDP, where the reduction rule is considered as the agent and remaining algorithmic procedures as the environment. At the t-th iteration, the state s_t is characterized by the current algorithmic status including the fixed information of the given MOIP instance; the current nondominated points N_t and their statistics; the found efficient solutions $\mathcal{X}_e^t$ and their statistics; the set of local upper bounds and the selected one u_t. The agent determines whether the selected local upper bound u_t is discarded (1) or not (0), i.e., $a_t \in \mathcal{A} = \{0, 1\}$, via the learned policy $\pi(a_t|s_t)$. With the decision from the agent, the environment transits from s_t to s_{t+1} by either solving the IP defined by u_t and updating the search region $S(N)$, or directly discarding u_t from $U(N)$. The above one-step transition iterates to yield an episode $\tau = (s_0, a_o, s_1, \ldots, s_T)$ with its probability $p(\tau) = p(s_0) \prod_{t=0}^{T-1} \sum_{a_t} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$. According to Proposition 2, the length of the trajectory (episode) is finite and generally larger than the cardinality of Pareto front due to the presence of futile IPs. The goal of MDP is to learn an optimal policy defined as below.

Definition 3. The optimal policy π^* for an ODA is the one finalizing, 1) N_T with all points in the Pareto front, and 2) the frequency of $a_t = 0$ in τ always equal to the cardinality of the Pareto front.

In other words, it would be ideal that with the learned reduction rule, every subroutine of the solver can produce one (effective) nondominated point, and all local upper bounds with their IPs resulting in infeasible or repetitive points are discarded with $\pi^*(a_t = 1|s_t) = 1$.

4.3 Graph learning

For graph encoding of states $\{s_t\}_{t=0}^{T-1}$ in the above MDP, it is crucial to represent multiple objective nodes for each respective MOIP as in Eq. (1) and identify varying scalarized problems (IPs) as in Eq. (2) along the solving process. To this end, we define a graph $\mathcal{G} = (\mathcal{C}, \mathcal{V}, \mathcal{O}, \mathcal{E}_{cv}, \mathcal{E}_{ov})$ with constraint nodes $\mathcal{C} = \{c_1, \cdots, c_m, c_{m+1}, \cdots, c_{m+p}\}$ featurized by a matrix $\mathbf{C} \in \mathbb{R}^{(m+p)\times d_c}$, variable nodes $\mathcal{V} = \{v_1, \cdots, v_n\}$ featurized by $\mathbf{V} \in \mathbb{R}^{n\times d_v}$, objective nodes $\mathcal{O} = \{o_1, \cdots, o_p\}$ featurized by $\mathbf{O} \in \mathbb{R}^{p\times d_o}$, edges $\mathcal{E}_{cv} = \{e_{ij}|c_i \in \mathcal{C}, v_j \in \mathcal{V}\}$ between constraint and variable nodes featurized by $\mathbf{E}_{cv} \in \mathbb{R}^{(m+p)\times n\times d_{cv}}$, edges $\mathcal{E}_{ov} = \{e_{kj}|o_k \in \mathcal{O}, v_j \in \mathcal{V}\}$ between objective and variable nodes featurized by $\mathbf{E}_{ov} \in \mathbb{R}^{p\times n\times d_{ov}}$. To derive accurate global representations of states, we further attach three hypernodes $\mathcal{H} = \{h_1, h_2, h_3\}$ to aggregate node representations from three partial graphs, that is, $\mathcal{G}_1 = (\mathcal{C}, \mathcal{V})$, $\mathcal{G}_2 = (\mathcal{O}, \mathcal{V})$ and $\mathcal{G}_3 = (\mathcal{C}, \mathcal{O})$. We illustrate the above graph in Figure 2 (the left one). The proposed graph encoding is generic for MOIPs, in which we can capture different scalarized problems by featurization (e.g. updating u for each IP). In this paper, we realize the featurization with respect to the Lexicographic scalarizations for the ODA [29], and the details are provided in Appendix A.4.

Two-stage GNN. Given the graph encoding, we propose a two-stage GNN (illustrated in the right half of Figure 2) to parameterize the reduction rule as $\pi_\theta(a_t|s_t)$. It pertains to the class of message-passing neural networks [43] with graph attention structure [44] and is enhanced by hypernode embedding which aggregates high-level graph information. In the first stage, the total four passes are executed along the two groups of undirected edges in the order $\mathcal{V} \to \mathcal{C} \to \mathcal{V} \to \mathcal{O} \to \mathcal{V}$, where the representations of target nodes for each pass are advanced as follows (taking $\mathcal{V} \to \mathcal{C}$ as an example),

$$
\begin{aligned}
\alpha_{ij} &= \frac{\text{SUM}\left(\text{LR}\left(\mathbf{v}_j W_v + \mathbf{e}_{ij} W_{cv} + \mathbf{c}_i W_c\right) \cdot W_a\right)}{\sum_{v_j \in \mathcal{V}} \text{SUM}\left(\text{LR}\left(\mathbf{v}_j W_v + \mathbf{e}_{ij} W_{cv} + \mathbf{c}_i W_c\right) \cdot W_a\right)}, \\
\mathbf{c}_i &= \text{MLP}_1((1+\varepsilon) \cdot \mathbf{c}_i + \text{MLP}_2(\sum\nolimits_{v_j \in \mathcal{V}} \alpha_{ij} \cdot \mathbf{v}_j)), \quad \forall i, c_i \in \mathcal{C}
\end{aligned}
\tag{3}
$$

where $\mathbf{c}_i$, $\mathbf{v}_j$ denote the i-th and j-th row of matrice $\mathbf{C}$ and $\mathbf{V}$ (i.e. features of c_i and v_j), and are separately projected into d_r-dimensional embeddings before feeding them into Eq. (3); $W_v, W_{cv}, W_c, W_a \in \mathbb{R}^{d_r \times d_r}$ are trainable parameters to compute the attention weight α_{ij}; LR and

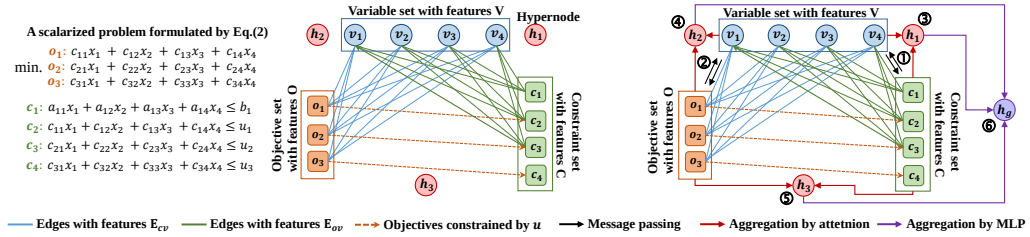

Figure 2: **An illustration of graph learning for MOIP instances with three objectives and one constraint.** The left subfigure shows the graph encoding for scalarized problems (IPs) in Eq. (2), where three constraints are varying since objectives are constrained by different local upper bound u along the solving process. The right subfigure shows the message flow with circled numbers, where ① and ② correspond to four message passes in the first stage among objectives, variables and constraints; ③ $\sim$ ⑥ correspond to the two-step aggregation in the second stage.

SUM mean Leaky ReLU activation and the addition of elements, respectively; MLP_1 and MLP_2 are two multilayer perceptrons, and each has two linear layers and one ReLU activation; ε could be a trainable parameter [40] and we set it to 0 in this paper. Following the above structure and logic, the representation of all nodes are updated after all passes are executed.

Since we aim to learn prediction for IPs that are encoded on graphs, an accurate representation of the entire graph is the foremost concern. Hence, in the second stage, we propose to aggregate the graph representation in a two-step manner as expressed in Eq. (4). Specifically, we first aggregate partial graphs ($\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$) on hypernodes (h_1, h_2, h_3) with attentions and then aggregate those hypernodes to attain the graph representation,

$$\begin{aligned} &\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3 = \text{Softmax}(\text{MLP}_3(\mathbf{H}_1))\mathbf{H}_1, \text{Softmax}(\text{MLP}_4(\mathbf{H}_2))\mathbf{H}_2, \text{Softmax}(\text{MLP}_5(\mathbf{H}_3))\mathbf{H}_3, \\ &h_g = \text{MLP}_6([\mathbf{h}_1; \mathbf{h}_2; \mathbf{h}_3]), \quad \text{given } \mathbf{H}_1 = [\mathbf{C}; \mathbf{V}], \mathbf{H}_2 = [\mathbf{O}; \mathbf{V}], \mathbf{H}_3 = [\mathbf{C}; \mathbf{O}], \end{aligned} \tag{4}$$

where $[;]$ means the concatenation operation; $\mathbf{h}_1$,$\mathbf{h}_2$ and $\mathbf{h}_3$ are representations of the three hypernodes; h_g denotes the score of the entire graph, meaning the extent that the current IP pertains to infeasibility or repetition; $\text{MLP}_3 \sim \text{MLP}_6$ refer to trainable scalar functions. We concretize $\text{MLP}_3 \sim \text{MLP}_5$ with two linear layers activated by ReLU function to compute attention weights, and concretize MLP_6 with two linear layers regularized by dropout in between. At last, we convert h_g with Sigmoid function into a value in $[0, 1]$, which represents the probability of the current IP in the state being infeasible or repetitive. We keep the hidden dimension $d_r = 128$ throughout the GNN.

4.4 Training algorithm

We train the policy network by behavioral cloning [45], a specific imitation learning algorithm, which has been widely used in policy optimization to solve single-objective IPs [19, 21, 24, 37]. We sample the state-action data in ODAs by using the optimal policy in Definition 3 as the oracle, and then train the policy network in a supervised fashion.

The collection of states and the optimal actions is summarized by the pseudocode in Appendix A.5. We run an ODA and label states causing infeasible or repetitive IPs by $a_t = 1$, meaning the optimal policy is supposed to discard the corresponding local upper bounds u_t without further solver calls. It is clear that the policy identified by the gathered state-action pairs is optimal for an MOIP instance according to Definition 3, since all nondominated points can be generated based on Proposition 2 and all infeasible or repetitive points are exactly avoided given $a_t = 1$. However, it would consume long time to collect all state-action pairs for an instance if it is complex or large-scale. In our training, we set a time limit L to collect our data, since we emphasize on the bounded-time performance, which is also important in practical applications. With the collected dataset $D = \{(s_i, a_i)\}_{i=1}^{I}$, we train the policy network by minimizing the binary cross-entropy loss defined as below,

$$\mathcal{L}(\theta) = -\mathbb{E}_D \, a_i \cdot \log \pi_\theta(a_i|s_i) + (1 - a_i) \cdot \log(1 - \pi_\theta(a_i|s_i)). \tag{5}$$

5 Experimental results

In the experiments, we deploy the proposed graph learning method on a state-of-the-art ODA [29], and evaluate it on two benchmark problems with varying numbers of objectives. We compare the learned algorithm with various baselines in literature, where multiple metrics are adopted. In addition, we assess the generalization of the learned reduction rules to MOIP instances which possess more objectives or larger sizes than the ones in training. Finally, we conduct the ablation study to verify the efficacy of the proposed two-stage GNN and the learned reduction rule.

5.1 Settings

We conduct experiments on two widely used benchmarks in the domain of MOIP, i.e., multi-objective knapsack problem (MOKP) and assignment problem (MOAP) [31], respectively.[4] Following the instance generation in [29], we uniformly sample the value and weight from $[1, 100]$ for each item in MOKP, and compute the capacity of the knapsack as half of the total weight. For MOAP, we randomly generate objective function coefficients (i.e. costs for each assignment) from $[1, 15]$.

Instances & datasets. For MOKP, we set the number of items to 100 and generate 110 instances with 3 and 4 objectives, respectively, with 100 instances for training and 10 for testing. We collect the training set by solving the 100 instances with the ODA [29] and Gurobi solver (v9.5.1) [46], as stated in Section 4.4. While we solve 3-objective instances optimally (i.e. attain all Pareto-optimal points), we set $L = 1000$s for 4-objective instances to collect the data so that we focus on training models for improving the bounded-time performance. We leave 1% data in the training set for validation. In the same way, the testing sets are collected by solving the 10 instances with 3 and 4 objectives, respectively. For MOAP, we set 50 agents and 50 tasks, respectively. We generate instances and collect the data in a similar way to MOKP except that we set $L = 1000$s for all instances. In summary, we train, validate and test the algorithms on MOKP and MOAP instances with 3 and 4 objectives, respectively. We name these instance groups as MOKP(3-100), MOKP(4-100), MOAP(3-50) and MOAP(4-50), with the first value in parentheses being the number of objectives and the second identifying the problem size. We also generate 10 instances for MOKP(4-150), MOKP(5-100), MOAP(4-75) and MOAP(5-50), respectively, to evaluate the generalization performance. Note that we set the number of objectives to the range $3\sim5$ as they are often used in ODAs [8, 30]. However, we also present more results with other numbers of objectives in Appendix A.6.

Training & testing. For training, we set the batch size to 64 and initial learning rate to 0.001. The learning rate decays by 0.2 once the validation loss is not decreased in 8 successive epochs. We employ early stopping to end the training when there is no improvement of validation performance in 16 successive epochs. We always retrieve the GNN model with the lowest validation loss along epochs. We employ Adam optimizer [47] to minimize the loss and execute the training with one GeForce RTX 2080 Ti GPU. For testing, we plug the learned GNN model in the state-of-the-art ODA [29], and use the GNN assisted algorithm to solve testing instances in different groups. In specific, we discard a local upper bound to save the solving of the corresponding IP if the output value from our GNN is larger than 0.5, so that the algorithm enters the next iteration by selecting another local upper bound, otherwise the algorithm will explore the IP by calling a solver (Gurobi in this paper) and update the search region as in the ODA. Except for GNN with GPU, the other algorithmic procedure is executed with one i9-10940X CPU@3.30GHz during testing.

5.2 Comparison study

We consider representative baselines of various paradigms for comparison, 1) three classic objective-space decomposition algorithms proposed by Kirlik and Sayın [31], Boland et al. [33], and Tamby and Vanderpooten [29] (the state-of-the-art, which is also used to deploy the proposed graph learning method), and we refer to them as ODA-K, ODA-B and ODA-T (it relies on Proposition 3 for infeasibility and solver for repetition), respectively; 2) the commonly used evolutionary algorithms including MOEAD, NSGAII, NSGAIII and UNSGAIII [48, 49, 50, 51]. We adapt and tune them for either of MOKP and MOAP [52, 53]; 3) the latest reinforcement learning model, i.e., PMOCO [10], which is specialized for MOCO problems with sequential structures. We adapt PMOCO to solve MOKP, which is trained on the same instances to ours, and other settings follow its original paper.

[4] http://home.ku.edu.tr/ moolibrary/

Table 1: Comparison with various baselines on MOKP and MOAP instances with 3 and 4 objectives.

	MOKP(3-100)				MOKP(4-100)				MOAP(3-50)				MOAP(4-50)			
Methods	Time(s)	Card.	HV	IGD	Time(s)	Card.	HV	IGD	Time(s)	Card.	HV	IGD	Time(s)	Card.	HV	IGD
ODA-T	993.3	**4169.8**	**0.55**	**0.000**	1000.0	3691.4	0.75	0.012	1000.0	2532.9	0.81	0.009	1000.0	2663.7	0.52	0.017
ODA-B	1401.2	**4169.8**	**0.55**	**0.000**	1000.0	3188.4	**0.76**	0.007	1000.0	1736.5	0.86	0.005	1000.0	1576.5	0.66	0.018
ODA-K	928.4	**4169.8**	**0.55**	**0.000**	1000.0	3091.8	0.61	0.055	1000.0	2952.2	0.58	0.022	1000.0	2443.0	0.24	0.049
MOEAD	2000.0	1234.2	0.52	0.026	2000.0	1224.3	0.73	0.060	2000.0	984.1	0.76	0.008	2000.0	2282.7	0.51	0.017
NSGAII	2000.0	578.3	0.41	0.007	2000.0	998.8	0.68	0.021	2000.0	150.6	0.52	0.020	2000.0	893.6	0.29	0.029
NSGAIII	2000.0	390.0	0.53	0.005	2000.0	188.2	0.69	0.022	2000.0	482.3	0.69	0.017	2000.0	294.2	0.50	0.029
UNSGAIII	2000.0	545.7	0.41	0.007	2000.0	262.7	0.68	0.026	2000.0	401.9	0.69	0.018	2000.0	333.5	0.48	0.031
PMOCO	1000.0	228.4	0.53	0.008	1000.0	632.0	0.75	0.014	-	-	-	-	-	-	-	-
Ours	**486.9**	3379.7	0.55	0.000	1000.0	**4000.7**	0.76	**0.006**	1000.0	**3564.4**	**0.87**	**0.004**	1000.0	**3771.3**	**0.67**	**0.011**

Table 2: Generalization to larger sizes and 5 objectives.

	MOKP(4-150)				MOKP(5-100)				MOAP(4-75)				MOAP(5-50)			
Methods	Time(s)	Card.	HV	IGD	Time(s)	Card.	HV	IGD	Time(s)	Card.	HV	IGD	Time(s)	Card.	HV	IGD
ODA-T	1000.0	3649.7	0.80	0.017	1000.0	2884.0	0.66	0.017	1000.0	1719.7	0.77	0.017	1000.0	2511.8	0.81	0.007
ODA-B	1000.0	2306.9	0.80	0.027	1000.0	2506.5	0.63	0.020	1000.0	899.1	0.73	0.037	1000.0	1312.9	0.75	0.028
ODA-K	1000.0	2588.0	0.53	0.109	1000.0	2654.2	0.45	0.110	1000.0	1640.3	0.24	0.064	1000.0	2236.9	0.36	0.047
MOEAD	2000.0	3834.1	0.75	0.089	2000.0	*4200.9*	0.60	0.084	2000.0	*2383.6*	0.45	0.037	2000.0	*3221.5*	0.59	0.026
NSGAII	2000.0	998.8	0.70	0.057	2000.0	1000.0	0.59	0.041	2000.0	930.7	0.24	0.062	2000.0	1105.7	0.34	0.043
NSGAIII	2000.0	185.6	0.72	0.058	2000.0	139.8	0.55	0.053	2000.0	209.9	0.46	0.062	2000.0	239.3	0.46	0.035
UNSGAIII	2000.0	264.5	0.71	0.069	2000.0	157.8	0.56	0.060	2000.0	233.2	0.40	0.056	2000.0	225.6	0.49	0.035
PMOCO	1000.0	732.0	0.81	0.023	1000.0	1475.5	0.66	0.020	-	-	-	-	-	-	-	-
Ours	1000.0	**4032.6**	**0.82**	**0.013**	1000.0	**3634.4**	**0.66**	**0.015**	1000.0	**2137.4**	**0.79**	**0.017**	1000.0	**3089.2**	**0.82**	**0.006**

We apply all methods on testing instances of MOKP(3-100), MOKP(4-100), MOAP(3-50) and MOKP(4-50), respectively. For MOKP(3-100), our method and three ODAs go through all local upper bounds since the instances are easy to be solved in relatively short time. For other instances, we set the time limit to 1000s for our method and ODAs. We set the same runtime for PMOCO across all instance groups and allow longer runtime (2000s) for evolutionary algorithms. To comprehensively assess the bounded-time performance, we adopt the following metrics, 1) Card.: cardinality of the returned set of nondominated points (for ODAs and ours) or feasible points (for others) [30];[5] 2) HV: hypervolume that reflects the convergence and spread of the approximated Pareto front [54]; 3) IGD: inverted generational distance that gauges the similarity between a set of images and the Pareto front [55]. We present details of the calculation for HV and IGD in Appendix A.7. For each metric, we record the average value across testing instances in each group and gather the results in Table 1.

For MOKP(3-100), we observe that the three classic ODAs attain all nondominated points and outperform other baselines in terms of the three metrics. However, our method is relatively fast to derive most nondominated points with less than half runtime. The nondominated points are very close to the Pareto front with almost the same HV and IGD as those in ODAs. In all approximate solutions, our method significantly surpasses evolutionary algorithms and PMOCO on all metrics. The advantage may stem from the facts that, 1) our method based on ODA-T adopts the highly-optimized IP solver (Gurobi) to fast compute nondominated points in scalarized problems, while solutions found in other heuristics have no guarantee to be nondominated; 2) the learned reduction rule further accelerates the search to find more nondominated points efficiently. We visualize the approximation (to Pareto front) by our method and heuristic baselines in Appendix A.8. Such advantage is also revealed on other instance groups, where our method consistently delivers higher HV and lower IGD than heuristic baselines. While ODA-T and ODA-B perform better than or comparable to the heuristics (in terms of HV and IGD) on MOKP(4-100), MOAP(3-50) and MOAP(4-50), our method generally outperforms the three ODAs on all metrics. On the other hand, although ODA-T has gained more nondominated points than ODA-B, ODA-K, our method with the learned reduction rule finds further more nondominated points than ODA-T in the same runtime, as shown in Card. columns. It

[5]While ODAs (thus also ours) ensure the optimal solutions to IPs are in the Pareto front according to Proposition 2, the other baselines belong to heuristic algorithms without any guarantee of solution quality.

Table 3: Ablation study on GNNs.

Models	MOKP 3-100	MOKP 4-100	MOAP 3-50	MOAP 4-50	MOKP 4-150	MOKP 5-100	MOAP 4-75	MOAP 5-50
GCN	68.4	67.6	72.7	91.2	**70.4**	58.8	94.5	90.6
GraphSage	63.0	70.8	69.6	91.2	59.5	59.9	94.8	**90.7**
GIN	79.7	82.0	79.6	91.2	59.5	58.8	93.4	90.6
2-stage GIN	80.7	**82.5**	80.8	**91.3**	59.5	58.8	**95.2**	90.6
Ours	**82.8**	80.9	**82.8**	**91.3**	68.6	**73.9**	95.0	**90.7**

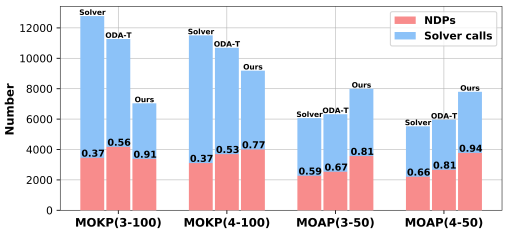

Figure 3: Ablation study on reduction rules.

indicates that our method is more effective in curtailing the solving of futile IPs, which could save runtime to explore more fruitful IPs. Meanwhile, the nondominated points found by our method show desirable convergence and spread, which are identified by higher HV and lower IGD values.

5.3 Generalization study

The generalization to instances larger than the ones in training is critical, as it may determine whether the learned algorithm could be used to solve a wider range of instances. Meanwhile, the generalization to more objectives is also valuable for MOIPs, where a decision maker may freely consider additional objectives through the learned algorithm. To this end, we apply the learned ODAs trained on MOKP(4-100) and MOAP(4-50) to solve instances with larger scales and more (i.e. 5) objectives, respectively. We also adapt evolutionary algorithms and PMOCO to solve these instances. Specially, we directly train PMOCO on 5-objective instances following [10], since it cannot generalize across objectives. All results are displayed in Table 2. It is manifested that our method attains the best values on all metrics for different problems. While MOEAD achieves significantly more feasible points and higher HV than other evolutionary algorithms and ODA-K, it performs inferior to PMOCO, ODA-T and ODA-B. Pertaining to ODAs, we observe that ODA-T generally outperforms ODA-B and ODA-K on all metrics and is comparable to PMOCO. However, our method with the learned reduction rule further improves ODA-T by circumventing futile IP solving, which also outstrips PMOCO. With the highest HV and lowest IGD, our method yields a better approximation to the Pareto front, implying that the learned reduction rule can generalize well to harder instances.

5.4 Ablation study

To evaluate the efficacy of our two-stage GNN, we compare it with the classic bipartite GCN [21], the latest bipartite GIN and GraphSage [39], all of which were used to solve IPs in their original papers. We arrange results in Table 3 where *test* and *generalization* accuracy are placed in the left and right half, respectively. We observe that our model generally transcends other GNNs for testing, except for MOKP(4-100) with a comparable accuracy to GIN. Meanwhile, our model delivers the highest generalization accuracy on most instance groups, except for MOKP(4-150), where it performs slightly inferior to GCN. We also verify the efficacy of our aggregation scheme (in the second stage) by equipping it to GIN given the better test performance than other baselines, which results in the two-stage GIN. As shown, the two-stage GIN improved the accuracy over GIN on $5/8$ instance groups, implying our scheme is effective to gain a better graph representation. However, our GNN still achieves superior overall performance to this enhanced GIN, suggesting that our attention in the first stage attains better node embeddings that benefit the subsequent graph representation learning.

Furthermore, we compare our method with ODA-T regarding the usage rate of the IP solver (Gurobi), to show the efficacy of the learned reduction rules. A vanilla version of ODA-T is also tested where we purely use solver to detect infeasibility and repetition (note: different from its vanilla version, ODA-T relies on Proposition 3 for infeasibility and solver for repetition). We plot the average numbers of nondominated points, solver calls and ratio in Figure 3. As shown, our method attains the most nondominated points on $3/4$ instance groups except for MOKP(3-100) which is easy for ODA-T to solve optimally. While the ratio of ODA-T is higher than its vanilla version, the learned reduction rule further enhances ODA-T with (at least) 13% higher ratio on different groups. It means the reduction rule assisted by our GNN is more effective than the one in ODA-T.

6 Conclusions and future work

This paper presents a graph neural network (GNN) based method to learn the reduction rule in an ODA, with the aim of improving its bounded-time performance in solving MOIPs. We train a novel two-stage GNN with imitation learning to discard local upper bounds that induce infeasibility and repetition of nondominated points. We implement our method with a state-of-the-art ODA and empirically show that it can significantly enhance the ODA and generalize to larger problems or more objectives. We also verify the superiority of the proposed GNN to existing ones for integer programs, in terms of better test and generalization accuracy. The limitation of our method may appear when the IP solver cannot fast solve intractable scalarized problems. A solution could be compromising the optimality in Proposition 1 and replace IP solver with heuristics, so that they could still be guided by the schemes in ODAs and the learned reduction rule. We leave this interesting direction as future work. Also, we plan to deploy our method to more ODAs.

Acknowledgments and Disclosure of Funding

This work was supported in part by the A*STAR Cyber-Physical Production System (CPPS) - Towards Contextual and Intelligent Response Research Program, under the RIE2020 IAF-PP grant A19C1a0018, and Model Factory@SIMTech; in part by the A*STAR HTPO seed grant C211118016, and the A*STAR Career Development Fund under grant C222812027; in part by the National Natural Science Foundation of China under grant 62102228, and the Shandong Provincial Natural Science Foundation under grant ZR2021QF063.

References

[1] Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer, 2014.

[2] Yunfei Cui, Zhiqiang Geng, Qunxiong Zhu, and Yongming Han. Multi-objective optimization methods and application in energy saving. *Energy*, 125:681–704, 2017.

[3] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[4] Naveen Saini and Sriparna Saha. Multi-objective optimization techniques: A survey of the state-of-the-art and applications. *The European Physical Journal Special Topics*, 230(10): 2319–2335, 2021.

[5] Jing Liang, Xuanxuan Ban, Kunjie Yu, Boyang Qu, Kangjia Qiao, Caitong Yue, Ke Chen, and Kay Chen Tan. A survey on evolutionary constrained multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2022.

[6] Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[7] José Rui Figueira, Carlos M Fonseca, Pascal Halffmann, Kathrin Klamroth, Luís Paquete, Stefan Ruzika, Britta Schulze, Michael Stiglmayr, and David Willems. Easy to say they are hard, but hard to see they are easy—towards a categorization of tractable multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 24(1-2):82–98, 2017.

[8] Pascal Halffmann, Luca E Schäfer, Kerstin Dächert, Kathrin Klamroth, and Stefan Ruzika. Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. *Journal of Multi-Criteria Decision Analysis*, 2022.

[9] Arne Herzel, Stefan Ruzika, and Clemens Thielen. Approximation methods for multiobjective optimization problems: A survey. *INFORMS Journal on Computing*, 33(4):1284–1299, 2021.

[10] Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022.

[11] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

[12] Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, pages 6278–6289, 2019.

[13] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[14] Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, and Xu Chi. Learning to dispatch for job shop scheduling via deep reinforcement learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[15] Fei Ni, Jianye Hao, Jiawen Lu, Xialiang Tong, Mingxuan Yuan, Jiahui Duan, Yi Ma, and Kun He. A multi-graph attributed reinforcement learning based optimization algorithm for large-scale hybrid flow shop scheduling problem. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3441–3451, 2021.

[16] Yuan Jiang, Zhiguang Cao, and Jie Zhang. Solving 3d bin packing problem via multimodal deep reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1548–1550, 2021.

[17] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):5057–5069, 2022.

[18] Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In *Advances in Neural Information Processing Systems*, volume 34, pages 11096–11107, 2021.

[19] He He, Hal Daume III, and Jason M Eisner. Learning to search in branch and bound algorithms. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NeurIPS)*, pages 3293–3301, 2014.

[20] Elias B Khalil, Bistra Dilkina, George L Nemhauser, Shabbir Ahmed, and Yufen Shao. Learning to run heuristics in tree search. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 659–666, 2017.

[21] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[22] Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement learning for integer programming: Learning to cut. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 9367–9376, 2020.

[23] Jialin Song, Ravi Lanka, Yisong Yue, and Bistra Dilkina. A general large neighborhood search framework for solving integer linear programs. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[24] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

[25] Yiyang Zhao, Linnan Wang, Kevin Yang, Tianjun Zhang, Tian Guo, and Yuandong Tian. Multi-objective optimization by learning space partitions. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022.

[26] Yinan Shao, Jerry Chun-Wei Lin, Gautam Srivastava, Dongdong Guo, Hongchun Zhang, Hu Yi, and Alireza Jolfaei. Multi-objective neural evolutionary algorithm for combinatorial optimization problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[27] Kaiwen Li, Tao Zhang, and Rui Wang. Deep reinforcement learning for multiobjective optimization. *IEEE transactions on cybernetics*, 51(6):3103–3114, 2020.

[28] Kathrin Klamroth, Renaud Lacour, and Daniel Vanderpooten. On the representation of the search region in multi-objective optimization. *European Journal of Operational Research*, 245 (3):767–778, 2015.

[29] Satya Tamby and Daniel Vanderpooten. Enumeration of the nondominated set of multiobjective discrete optimization problems. *INFORMS Journal on Computing*, 33(1):72–85, 2021.

[30] Özlem Karsu and Firdevs Ulus. Split algorithms for multiobjective integer programming problems. *Computers & Operations Research*, 140:105673, 2022.

[31] Gokhan Kirlik and Serpil Sayın. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232 (3):479–488, 2014.

[32] Kerstin Dächert, Kathrin Klamroth, Renaud Lacour, and Daniel Vanderpooten. Efficient computation of the search region in multi-objective optimization. *European Journal of Operational Research*, 260(3):841–855, 2017.

[33] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. *European journal of operational research*, 260(3):904–919, 2017.

[34] Miguel Ángel Domínguez-Ríos, Francisco Chicano, and Enrique Alba. Effective anytime algorithm for multiobjective combinatorial optimization problems. *Information Sciences*, 565: 210–228, 2021.

[35] Tim Holzmann and J Cole Smith. Solving discrete multi-objective optimization problems using modified augmented weighted tchebychev scalarizations. *European Journal of Operational Research*, 271(2):436–449, 2018.

[36] Jianya Ding, Chao Zhang, Lei Shen, Shengyin Li, Bing Wang, Yinghui Xu, and Le Song. Accelerating primal solution findings for mixed integer programs based on solution prediction. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

[37] Prateek Gupta, Maxime Gasse, Elias Khalil, Pawan Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*, 33:18087–18097, 2020.

[38] Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Learning large neighborhood search policy for integer programming. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, volume 34, pages 6278–6289, 2021.

[39] Elias B Khalil, Christopher Morris, and Andrea Lodi. Mip-gnn: A data-driven framework for guiding combinatorial solvers. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, 2022.

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

[41] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pages 1025–1035, 2017.

[42] Matthias Ehrgott. A discussion of scalarization techniques for multiple objective integer programming. *Annals of Operations Research*, 147(1):343–360, 2006.

[43] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1263–1272, 2017.

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[45] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[46] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL https://www.gurobi.com.

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[48] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[49] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6 (2):182–197, 2002.

[50] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.

[51] Haitham Seada and Kalyanmoy Deb. A unified evolutionary optimization procedure for single, multiple, and many objectives. *IEEE Transactions on Evolutionary Computation*, 20(3):358–369, 2015.

[52] Robert F Subtil, Eduardo G Carrano, Marcone JF Souza, and Ricardo HC Takahashi. Using an enhanced integer nsga-ii for solving the multiobjective generalized assignment problem. In *IEEE congress on evolutionary computation*, pages 1–7, 2010.

[53] Thibaut Lust and Jacques Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *International Transactions in Operational Research*, 19(4): 495–520, 2012.

[54] Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 862–876. Springer, 2007.

[55] Peter AN Bosman and Dirk Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE transactions on evolutionary computation*, 7(2):174–188, 2003.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [N/A]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 3 and 4.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 5.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.1.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Graph Learning Assisted Multi-objective Integer Programming (Appendix)

A.1 Search region update

According to Definition 2 in the main paper, the search region is described by a set of (maximal) local upper bounds. Hence, we update the search region by efficiently updating the local upper bounds when a new nondominated point is attained based on Eq. (2). We summarize the update procedures in Algorithm 1 as below. It is originally proposed in [26] and used by the ODA in [27], on which we deploy the graph learning method.

Algorithm 1: Update of local upper bounds

1: **Input:** A set of points N, the corresponding local upper bounds $U(N)$, a new nondominated point y^* (Note: $U(N)$ is initialized as $U(\emptyset) = \{(M, \dots, M)\}$ as introduced in the main paper)
2: $N' = N \cup y^*; U(N') = U(N)$;
3: **for** $u \in U(N)$ **do**
4: ## if y^* dominates u, u is deleted and new bounds are added; or, y^* could be a defining point of u. ##
5: **if** $y^* < u$ **then**
6: $U(N') = U(N') \backslash \{u\}$;
7: **for** $i \in \{1, \dots, p\}$ **do**
8: $u^i = (u_1, \dots, u_{i-1}, y_i^*, u_{i+1}, \dots, u_p)$;
9: $N_i(u^i) = \{y^*\}$
10: **for** $j \in \{1, \dots, p\} \backslash \{i\}$ **do**
11: $N_j(u^i) = \{y | y \in N_j(u), y_i < y_i^*\}$
12: **if** $N_j(u^i) \neq \emptyset, \forall \{j | j \in \{1, \dots, p\} \backslash \{i\}, u_j^i \neq M\}$ **then**
13: $U(N') = U(N') \cup \{u^i\}$
14: **else**
15: **for** $i \in \{1, \dots, p\}$ **do**
16: **if** $y_i^* = u_i$ and $y_{-i}^* < u_{-i}$ **then**
17: $N_i(u) = N_i(u) \cup \{y^*\}$
18: **Return:** updated local upper bounds $U(N')$

In the above algorithm, $N_i(u)$ denotes the defining points for the i-th dimension of the local upper bound u. It satisfies, 1) $N_i(u) \subset N$, and 2) $\forall y \in N_i(u)$, $y_k = u_k$ and $y_{-k} < u_{-k}$, where y_{-k} refers to a reduction of y with its k-th component eliminated. According to [26, 27], the set of local upper bounds is maximal if and only if every bound in the set has at least one defining point on the dimensions $\{i | i \in \{1, \dots, p\}, u_i \neq M\}$. Therefore, Algorithm 1 adds new local upper bounds by ensuring they have defining points for bounded dimensions (line 7 $\sim$ 13). When $y_i^* = u_i$ and $y_{-i}^* < u_{-i}$, y^* is exactly a defining point of u and added to $N_i(u)$ (line 15 $\sim$ 17). We refer the readers to [26, 27] for more details.

A.2 Scalarizations

ODAs normally rely on scalarization methods to convert MOIP into a series of single-objective IPs. A variety of scalarization methods have been proposed for different ODAs. Given a MOIP as in Eq. (1) in the main paper, we introduce four commonly used methods including the one (i.e. Lexicographic optimization) used in our approach. More scalarization methods can be found in [6, 8].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Weighted sum method. It minimizes a weighted sum of objectives as follows: $\min_x\{\sum \mu_i f_i(x)|\mu \in \mathbb{R}^p; x \in \mathcal{X}\}$. The optimal solution to such scalarization is an (weakly) efficient solution to the MOIP when μ $(\geq)>0$. While this method is simple and widely used, it is not appropriate for nonconvex objective spaces since some nondominated points cannot be represented by the scalarized objective.

ϵ-constraint method. It optimizes one objective with constraints to other objective values as follows: $\min_x\{f_k(x)|f_i(x) \leqslant \epsilon_i, i \neq k, \epsilon \in \mathbb{R}^p; x \in \mathcal{X}\}$. Its optimal solution is weakly efficient for MOIP, and many variants have been proposed to derive efficient solutions. Classic ϵ-constraint methods enumerate ϵ to compute all nondominated points and require considerable iterations of IP solving when the number of objectives rises. ODAs can be used as a paradigm to improve the classic methods since they efficiently exclude the dominated region by newly derived points in the objective space.

Weighted Tchebycheff method. It minimizes the maximal distances between objectives and the ideal point $y^I \in \mathbb{R}^p$ as follows: $\min_x\{\max_{1\leqslant i\leqslant p}\{\mu_i(f_i(x) - y_i^I)\}|x \in \mathcal{X}\}$, where $\mu \geq 0$ captures the preference over different objectives. The optimal solution to the above formulation is weakly efficient to MOIP, and variants have been developed to compute efficient solutions, e.g., lexicographic and augmented weighted Tchebycheff methods.

Lexicographic optimization method. It is used in the ODA [27], which is defined as follows: $\text{lexmin}\{\{f_k(x), \sum_{i=1,i\neq k}^{p} f_i(x)\}|f_i(x) \leqslant u_i, i \neq k, u \in \mathbb{R}^p; x \in \mathcal{X}\}$. The k-th objective is optimized in priority and the sum of the other objectives is optimized without impairing the optimal value of the k-th objective. Similar to the ϵ-constraint method, u is used to restrict the objective values, i.e., the search region for the above IP in the objective space.

A.3 Proofs

Here we rewrite Proposition 1 and 2 and present proofs for them.

Proposition 1. Given $u \in U(N)$ and a strictly monotone function $\mathcal{F} : \mathcal{Y} \to \mathbb{R}$, the optimal solution to the following scalarized problem (IP) will derive a nondominated point in $\mathcal{Y}_{nd}$,

$$\Gamma(u) := \min\{\mathcal{F}(y) = \mathcal{F}(y_1, \ldots, y_p)|y \in \mathcal{Y}; y < u\}. \tag{A.1}$$

Proof. We assume the optimal solution y^* to $\Gamma(u)$ is not a nondominated point, i.e., $y^* \notin \mathcal{Y}_{nd}$. Therefore, there exists a point $y' \in \mathcal{Y}_{nd}$ satisfying $y' \leq y^* < u$. According to the strictly monotone property of $\mathcal{F}$, we have $\mathcal{F}(y') < \mathcal{F}(y^*)$ which contradicts with the optimality of y^* in $\Gamma(u)$.

Proposition 2. ODAs generate the whole Pareto front with finite iterations.

Proof. Given the bounded MOIP defined in Eq. (1) in the main paper, the number of nondominated points is finite, i.e., $|\mathcal{Y}_{nd}| < \infty$. The ODAs iteratively solve IPs with the form in Eq. (A.1), and each arrives at the following possible results: 1) a new nondominated point, 2) a repetitive nondominated point, and 3) infeasibility. According to Definition 2, the remaining nondominated points are located in the search region delimited by the local upper bounds. Thus solving IPs (constrained by the bounds) optimally in iterations will attain all nondominated points, according to Proposition 2. Meanwhile, the local upper bounds inducing infeasible IPs and repetitive points are ignored by ODAs, which means the regions without nondominated points are excluded. Hence, the number of iterations cannot be infinite within the bounded and discrete objective space.

A.4 Featurization

Given the graph encoding in the main paper, we can readily represent different scalarized problems by different featurizations on the graph. Here, we realize the featurization of the Lexicographic scalarizations for the ODA in [27], since we use it to deploy our graph learning method. According to the original work in [27], the Lexicographic scalarization is defined as follows,

$$\begin{aligned} \text{lexmin} \quad & \left\{ f_k(x), \sum_{i=1,i\neq k}^{p} f_i(x) \right\} \\ \text{s.t.} \quad & f(x) \in \mathcal{Y}, \;\; f_i(x) \leqslant u_i, \forall i \in \{1, \ldots, p\}, i \neq k \end{aligned} \tag{A.2}$$

Table A.1: Featurization of the Lexicographic scalarization on the graph.

Features	Description	Digit
Objective features ($\mathbf{O}$)	The ideal point $y_i^I = \min_{x \in \mathcal{X}} f_i(x)$.	1
	The selected local upper bound u.	1
	The defining point as described in Section A.1.	1
	The average, maximum and minimum of current local upper bounds.	3
	One-hot vector to identify which objective is in priority.	1
Constraint features ($\mathbf{C}$)	The right-hand side of constraints in Eq. (A.2).	1
Variable features ($\mathbf{V}$)	The average and standard deviation of history solutions.	2
	Coefficients of variables for the objective in priority.	1
Edge features ($\mathbf{E}_{ov}$)	Coefficients of variables for all objectives.	1
Edge features ($\mathbf{E}_{cv}$)	Coefficients in the incidence matrix between variables and constraints.	1

where $u = (u_1, \ldots, u_p)$ is the selected local upper bound in an iteration of ODA. The logic behind the above Lexicographic scalarization is that the k-th objective is optimized in priority, so that the sum of the remaining objectives are optimized without impairing the optimized value of the k-th objective. The ODA in [27] employs IP solver to solve the above scalarized problems (IPs) efficiently.

Our work centers on learning representations of scalarized problems in the above form to predict whether the corresponding IPs 1) are infeasible, or 2) lead to repetition of nondominated points, with the proposed two-stage GNN. Based on the prediction, we choose to solve the scalarized problem, or discard the selected local upper bound and enter the next iteration. To effectively capture the above scalarized problem, we leverage a set of features as described in Table A.1.

A.5 Pseudocode for dataset collection

Algorithm 2: Data collection for a MOIP instance with an ODA

1: **Input:** MOIP instance z, local upper bounds $U = U(\emptyset)$, nondominated points $N = \{\}$, time limit L, step $t = 0$, dataset $D_z = \{\}$
2: **while** runtime $< L$ **do**
3: Select a local upper bound u_t from U;
4: Record the current state s_t by collecting the features in Section A.4;
5: Solve the IP defined by u_t with the solver;
6: **if** IP is infeasible or returned point y is already in N **then**
7: $D_z = D_z \cup (s_t, 1)$; Discard u_t from U;
8: **else**
9: $D_z = D_z \cup (s_t, 0)$; Update U by ODA itself as described in Section A.1; $N = N \cup y$
10: **if** U={} **then**
11: **Return:** D_z
12: $t = t + 1$;
13: **Return:** D_z

According to the optimal policy in Definition 3 in the main paper, we collect states and the optimal actions for each instance following the procedure in Algorithm 2. We run an ODA and label states which induce infeasible IPs or repetitive nondominated points, by $a_t = 1$. The proposed graph learning approach aims to approximate such an optimal policy by discarding the local upper bounds u_t in futile IPs. We execute Algorithm 2 for every training instance in each group and collect the training set $D = \bigcup_{z=1}^{Z} D_z = \{(s_i, a_i)\}_{i=1}^{I}$. Except for the component we plug in to extract features, the other components remain the same as in the original ODA, such as the selection of the bounds at line 3 and the update of the search region at line 9.

A.6 Additional results

We present more results on instances with 6 and 7 objectives. Following the instance generation in the main paper, we generate 110 instances of MOKP(6-100) and MOAP(6-40), respectively,

Table A.2: Comparison with baselines on MOKP and MOAP instances with 6 and 7 objectives.

	MOKP(6-100)				MOKP(7-100)				MOAP(6-40)				MOAP(7-50)			
Methods	Time(s)	Card.	HV	IGD	Time(s)	Card.	HV	IGD	Time(s)	Card.	HV	IGD	Time(s)	Card.	HV	IGD
ODA-T	1000.0	2554.6	0.53	0.059	1000.0	1946.3	0.45	0.080	1000.0	1976.3	0.76	0.016	1000.0	951.5	0.80	0.024
ODA-B	1000.0	1684.2	0.53	0.064	1000.0	1099.6	0.49	0.081	1000.0	1509.8	0.73	0.019	1000.0	980.9	0.83	0.023
ODA-K	1000.0	2437.1	0.40	0.075	1000.0	2025.9	0.31	0.104	1000.0	1966.5	0.35	0.037	1000.0	1228.5	0.45	0.051
MOEAD	2000.0	*3612.5*	0.50	0.061	2000.0	*4345.1*	0.46	0.072	2000.0	*2958.9*	0.63	0.021	2000.0	*3235.4*	0.67	0.033
NSGAII	2000.0	1000.0	0.47	0.075	2000.0	1000.0	0.40	0.065	2000.0	1793.5	0.35	0.035	2000.0	2000.0	0.43	0.052
NSGAIII	2000.0	140.2	0.44	0.084	2000.0	150.3	0.35	0.090	2000.0	241.1	0.44	0.029	2000.0	243.1	0.49	0.045
UNSGAIII	2000.0	149.7	0.45	0.085	2000.0	121.8	0.38	0.099	2000.0	235.9	0.45	0.028	2000.0	237.7	0.52	0.044
PMOCO	1000.0	2464.7	0.53	0.061	1000.0	2688.5	**0.51**	**0.064**	-	-	-	-	-	-	-	-
Ours	1000.0	2996.4	**0.54**	**0.057**	1000.0	2392.8	0.51	0.065	1000.0	2261.3	**0.80**	**0.009**	1000.0	1358.2	**0.87**	**0.012**

with 100 instances for the collection of training set (in which 1% is used for validation) and 10 instances for testing. We also generate 10 instances of MOKP(7-100) and MOAP(7-50) to evaluate the generalization of the models trained with MOKP(6-100) and MOAP(6-40), respectively. We train PMOCO with instances of MOKP(6-100) and MOKP(7-100) since it cannot generalize across different numbers of objectives. The results are displayed in Table A.2.

We observe that our method achieves the highest HV and the lowest IGD values for MOKP(6-100), MOAP(6-40) and MOAP(7-50). While it performs slightly inferior to PMOCO for MOKP(7-100), i.e., 0.508/0.512 w.r.t HV and 0.065/0.064 w.r.t IGD, we note that our model is directly applied to instances with more objectives and PMOCO has no such generalization capability. Particularly, without the need of extra efforts to train the model with new data, our method attains comparable results to PMOCO. This advantage allows our method to be more preferable in practice where a decision maker can add objectives freely and get the results quickly. Furthermore, our method significantly improves ODA-T with more nondominated points found in the same runtime, and the cardinality is also larger than those of ODA-B, ODA-K and most heuristic baselines. While MOEAD achieves the largest cardinality in each instance group and mostly attains the best performance among evolutionary algorithms, it is still inferior to ODA-T, ODA-B and our method. This stems from the fact that the (feasible) solutions returned by MOEAD are of poor quality and ODAs (including ours) guarantee that the solutions are exactly nondominated. In summary, our method not only achieves the best performance for instances that are similar to the ones in training, but also generalizes fairly well to instances with more objectives and larger sizes.

A.7 Computation of HV and IGD

Hypervolume. The hypervolume for a set of feasible points F is defined as the volume of a subspace in the objective space, which is weakly dominated by F and bounded by a reference point r^*, as follows,

$$\text{HV}(F) = \lambda^p(\{y \in \mathbb{R}^p | \exists y' \in F, y' \leq y \leq r^*\}), \tag{A.3}$$

where λ^p denotes the Lebesgue measure on the p-dimensional space which calculates the volume for a p-dimensional subspace. The hypervolume is a widely used metric to assess the performance of approximated solutions. Since the range of objective values varies across different problems and instances, we employ the normalized hypervolume (NHV) following [10], which is defined as $\text{NHV}(F) = \text{HV}(F)/\prod_{i=1}^{p} r_i^*$. We compute r^* for an instance as element-wise maximums of all points from different methods, so that NHVs of solutions from each method are derived for this instance. We then compute the average NHV across the instances used for testing.

Inverted generational distance. Given a set of feasible points F and the reference set Y, the inverted generational distance is defined as follows,

$$\text{IGD}_\sigma(Y, F) = \frac{1}{|Y|} \Big(\sum_{y \in Y} \min_{f \in F} d(y, f)^\sigma \Big)^{1/\sigma}, \tag{A.4}$$

where $d(y, f) = (\sum_{i=1}^{p} (y_i - f_i)^2)^{1/2}$ represents Euclidean distance and σ is set to 1. The reference set is set as the Pareto front if all nondominated points are known. However, it is often not practical

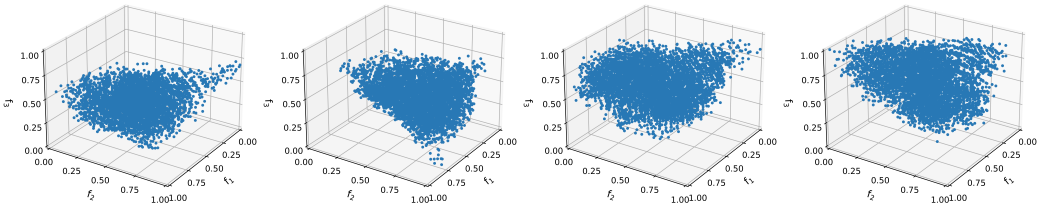

Figure A.1: The exact Pareto front.

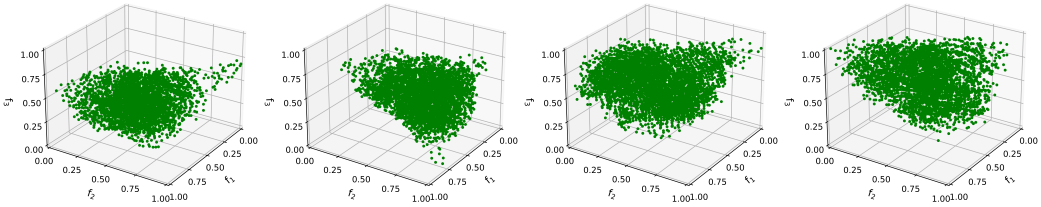

Figure A.2: Approximated Pareto front by our method.

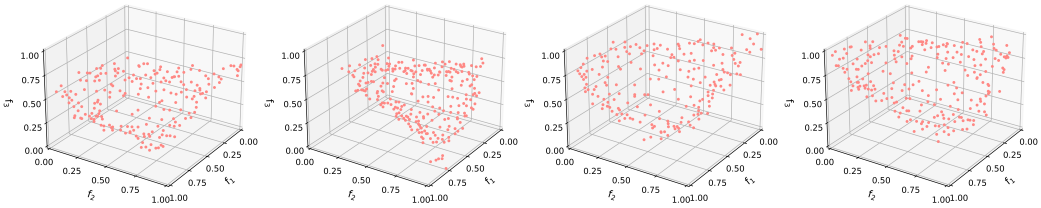

Figure A.3: Approximated Pareto front by PMOCO.

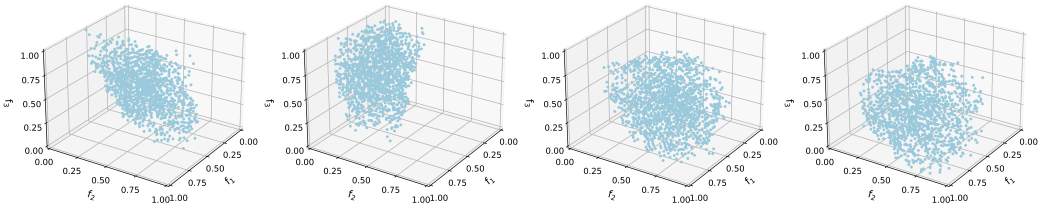

Figure A.4: Approximated Pareto front by MOEAD.

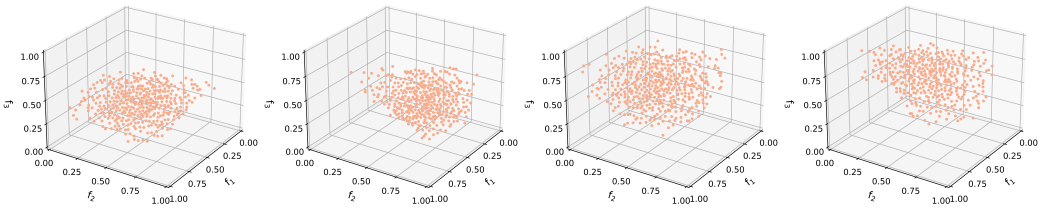

Figure A.5: Approximated Pareto front by NSGAIII.

for complex MOIPs in which the whole Pareto front is intractable to compute in prior. On the other hand, the reference set could also be a (good) approximated Pareto front for assessment. In this paper, we use the exact Pareto front for MOKP(3-100) to compute IGD since they are easy to be solved optimally. For the other instances, we gather all nondominated points from all methods as the good approximated Pareto front. This approximation is closer to the true Pareto front and thus better than the approximated solutions from each method. Moreover, we find that it is better than the one achieved by running a single method with long time, since different methods could generate different parts of nondominated points in Pareto front and we gather them together to gain better spread.

A.8 Visualization

We visualize the solutions attained by different methods. We pick four representatives from all baselines in the main paper including ODA-T in ODAs, MOEAD, NSGAIII in evolutionary computation, and the recent PMOCO in deep reinforcement learning. Among them, ODA-T achieves the exact Pareto front. Regarding each baseline, we plot solutions to four testing instances for MOKP(3-100) by normalizing the objective values to $[0, 1]^{\mathbb{R}=3}$, which are displayed in Figure A.1$\sim$ A.5. It is clear that the solutions achieved by our method is almost the same to the exact Pareto front, with good convergence and spread. In comparison, MOEAD and NSGAIII exhibit clear biases towards only parts of the Pareto front. While PMOCO scatters its points dispersedly with a similar shape to the Pareto front, the points are too sparse to approximate the details. Furthermore, our method guarantees the derived points are exactly nondominated (i.e. in Pareto front) and thus gains a substantially better approximation. Last but not least, our method consumes significantly less runtime than other methods (i.e. less than half) as stated in the main paper, which suggests a favorable trade-off to fast achieve an adequately good approximated Pareto front.