DiBS−MTL: TRANSFORMATION-INVARIANT MULTITASK LEARNING WITH DIRECTION ORACLES

Anonymous authors
Paper under double-blind review

ABSTRACT

Multitask learning (MTL) algorithms typically rely on schemes that combine different task losses or their gradients through weighted averaging. These methods aim to find Pareto stationary points by using heuristics that require access to task loss values, gradients, or both. In doing so, a central challenge arises because task losses can be arbitrarily, nonaffinely scaled relative to one another, causing certain tasks to dominate training and degrade overall performance. A recent advance in cooperative bargaining theory, the Direction-based Bargaining Solution (DiBS), yields Pareto stationary solutions immune to task domination because of its invariance to monotonic nonaffine task loss transformations. However, the convergence behavior of DiBS in nonconvex MTL settings is currently not understood. To this end, we prove that under standard assumptions, a subsequence of DiBS iterates converges to a Pareto stationary point when task losses are possibly nonconvex, and propose DiBS−MTL, a computationally efficient adaptation of DiBS to the MTL setting. Finally, we validate DiBS−MTL empirically on standard MTL benchmarks, showing that it achieves competitive performance with state-of-the-art methods while maintaining robustness to nonaffine monotonic transformations that significantly degrade the performance of existing approaches, including prior bargaining-inspired MTL methods.

1 INTRODUCTION

The successes of deep learning have inspired investigation into "generalist" networks—models simultaneously trained for learning multiple tasks. As a result, numerous multitask learning (MTL) algorithms have been developed to tackle the inevitable conflict between task-specific loss gradients, aiming to ensure that during training, no task is under-optimized compared to others (Kendall et al., 2018; Sener & Koltun, 2018; Yu et al., 2020a; Liu et al., 2021a; Navon et al., 2022; Liu et al., 2023). However, most existing MTL methods are not robust against non-affine (monotonic) transformations to task losses, which is a crucial property desirable in the context of deep learning—where same preferences can be represented with losses of different nonaffine scalings, and it is unclear which relative scaling of the different losses ensures balanced learning without expensive and exhaustive ablations. We consider the problem of multitask learning (MTL) through the lens of centralized, cooperative bargaining methods that are invariant to *non-affine* monotonic task loss transformations.

The issue of different task losses being directly incomparable and scaled in different, non-affine fashions arises very naturally in many deep learning domains. For instance, reinforcement learning applications demand that a practitioner leverages prior, task-specific domain knowledge to design an effective reward function (Yu et al., 2025). However, in a downstream MTL setting, the loss corresponding to this reward function may dominate (or get dominated by) other task losses. At the same time, the relative performance of a "good" task-specific policy does not change when the corresponding task loss is monotonically transformed,i.e., the transformation does not change the actual underlying preferences over options. *Thus, in an MTL problem, the available task losses can be seen as monotonic—possibly non-affine—transformations of some underlying set of ideal, unknown task losses that are meaningfully scaled with respect to each other.*

Recent work in MTL has developed a connection with cooperative game theory (Navon et al., 2022). In this setting, each different loss function is a separate player in a bargaining game, and the idea is to find a balanced Pareto optimum among the players' objectives. Classical solutions to these bargain-

ing games (e.g., Nash, as explored by Navon et al. (2022)) are *not* robust to non-affine monotonic scalings, and only recently has a technique—Direction-based Bargaining Solution (DiBS)—been developed which remains *invariant* to these transformations (Gupta et al., 2025). However, the convergence of DiBS has only been analyzed in settings with strongly convex losses, and it is unclear to what extent the favorable properties of DiBS will apply in realistic MTL applications where task losses are almost always nonconvex. Inspired by this, we investigate the following:

1. *What theoretical properties can be established for Direction-based Bargaining Solution (DiBS) in the general setting where player objectives (task losses) can be non-convex?*

   Contribution 1. We show that under standard assumptions, for non-convex losses, a subsequence of the DiBS iterates provably converges to a Pareto stationary point asymptotically. Notably, our result does not require the linear independence of task loss gradients at non-Pareto stationary points, an assumption that is required by MTL methods using the Nash bargaining solution to deliver the same asymptotic guarantee (Navon et al., 2022).

2. *Can DiBS readily adapt to MTL applications? If so, how does it compare to existing MTL methods?*

   Contribution 2. We extend DiBS to multitask learning, showing its natural compatibility with existing bargaining-for-learning frameworks. Moreover, we propose an approximation, DiBS-MTL, which is computationally more efficient than DiBS, and also preserves desirable invariance to non-affine monotonic task loss transformations. We empirically show that DiBS-MTL performs competitively with existing MTL methods on widely used multitask computer vision and reinforcement learning benchmarks.

   Contribution 3. We further investigate multitask reinforcement learning settings in which different task reward functions undergo non-affine monotonic transformations—causing potentially *non-monotonic* critic loss transformations. We empirically demonstrate that in this setting, DiBS-MTL maintains robust performance, while existing state-of-the-art MTL methods observe a significant drop in performance under the same transformations.

2 On related MTL works and existing bargaining solutions

2.1 Related MTL literature

The most popular MTL approach in practice is linear scalarization (LS)—constructing a scalarized loss by taking the unweighted sum of task losses, or using known static coefficients to compute a weighted average. While previous work has advocated for LS methods (Kurin et al., 2022; Xin et al., 2022), in practice, LS can lead to situations where certain tasks remain under-optimized. Further, it has been shown that LS also does not necessarily recover the entire Pareto front generated by the task losses (Hu et al., 2023). Other methods tackle the MTL problem through more sophisticated multiobjective optimization approaches, aiming to find Pareto optimal (or stationary, in general non-convex settings) points. Such methods seek to address the task imbalances arising during training via heuristics that (i) use task-specific loss values to compute a scalar weighted average loss (but with evolving weights, unlike LS) (Kendall et al., 2018; Liu et al., 2019; Lin et al., 2022; Liu et al., 2023), or (ii) use task-specific loss gradients to find update directions iteratively during training (Sener & Koltun, 2018; Yu et al., 2020a; Chen et al., 2020; Liu et al., 2021a;b; Navon et al., 2022).

Existing loss-based heuristics include maximizing improvement of the worst-performing task (Liu et al., 2023), forcing improvements to be similar across tasks (Liu et al., 2019), weighting task losses randomly (Lin et al., 2022), and adapting weights according to task-based uncertainty measures (Kendall et al., 2018). Heuristics for gradient-based approaches include finding mutual task improvement directions (Yu et al., 2020a), probabilistically masking task gradients according to their sign (Chen et al., 2020), computing weights that minimize the norm of the convex combination of task gradients (Sener & Koltun, 2018), using gradient-based approaches to maximize improvement of the worst-performing task (Liu et al., 2021a), and finding the Nash bargaining solution for a bargaining subproblem (Navon et al., 2022). However, as we empirically demonstrate in Section 5, these *existing methods are not robust to monotonic, nonaffine task loss transformations*, potentially experiencing significant performance drops in such settings. We remark than one gradient-based method, IMTL-G (Liu et al., 2021b), in principle can produce solutions invariant to monotonic non-

affine task loss transformations. However, we show in Appendix D that even for simple convex problems, IMTL-G can converge to Pareto solutions that heavily favor one task over the other.

Finally, we note that other approaches also exist for the MTL problem, such as (i) task clustering, where methods first cluster tasks to reduce conflicts, and then update model training parameters for each cluster (Standley et al., 2020; Fifty et al., 2021; Song et al., 2022; Shen et al., 2024); and (ii) parameter sharing methods, which design neural network architectures consisting of task-specific and task-shared modules/parameters (Kokkinos, 2017; Guo et al., 2020; Gao et al., 2020). These approaches differ from the proposed approach at a fundamental level, in that they try to design a suite of models for the space of tasks, or design model architectures suitable for MTL, whereas the proposed approach aims to optimize a *a single* model (with a pre-defined architecture), which balances the performance of all tasks.

2.2 Preliminaries on cooperative bargaining games

We provide a brief background of cooperative bargaining theory, which we utilize in our main contribution. A thorough description can be found in classical literature (Thomson, 1994; Narahari, 2014). A centralized bargaining game consists of N agents and a mediator; and $\mathbf{x} \in \mathcal{S} \subseteq \mathbb{R}^n$ denotes the state of the game. We assume the i^{th} agent has a differentiable cost $\ell^i(\mathbf{x}) : \mathcal{S} \to \mathbb{R}, i \in [N]$. In the context of MTL, every task can be thought of as being represented by one agent, with the agent's cost being the task loss.

Each agent want's to minimize its cost, and has preferred states $\mathbf{x}^{*,i} \in \arg\min_{\mathbf{x}\in\mathcal{S}} \ell^i(\mathbf{x})$ it wants the game to go towards. For nonconvex costs, this could be a local minimum. The goal of the mediator is to execute a bargaining strategy and find a solution state $\mathbf{x}^\dagger$. Let $\boldsymbol{\ell}(\mathbf{x}) = [\ell^i(\mathbf{x}), \ldots, \ell^N(\mathbf{x})]$. For convenience, we denote such a bargaining game by $\mathcal{B}_\mathcal{S}(\boldsymbol{\ell})$. Numerous bargaining solutions have been proposed in economics literature (Thomson, 1994), each employing different heuristics to find a Pareto optimal point—a point at which no update direction exists, which simultaneous decreases the losses for all agents. In the case when the agent losses $\ell^i(\mathbf{x})$ are nonconvex, a more appropriate goal for the mediator is to find a *Pareto stationary* point, which is a first-order necessary condition for Pareto optimality.

Definition 1 (**Pareto stationarity**). For $\mathcal{B}_\mathcal{S}(\boldsymbol{\ell})$, a point $\mathbf{x}^\dagger \in \mathcal{S}$ is Pareto stationary if $\exists\ \beta^i \geq 0, i \in [N]$, such that $\sum_{i\in[N]} \beta^i \nabla_\mathbf{x} \ell^i(\mathbf{x}^\dagger) = 0$, and $\sum_{i\in[N]} \beta^i = 1$.

We now highlight the Direction-based Bargaining Solution (DiBS), which we use in our method. For a bargaining game $\mathcal{B}_\mathcal{S}(\boldsymbol{\ell})$, DiBS finds a Pareto stationary point by taking an initial point $\mathbf{x}_1 \in \mathcal{S}$ and running the iterations

$$\mathbf{x}_{k+1} = \mathbf{x}_k - h(\mathbf{x}_k) := \mathbf{x}_k - \alpha_k \cdot \sum_{i\in[N]} \left(\|\mathbf{x}_k - \mathbf{x}^{*,i}\|_2 \cdot \frac{\nabla_\mathbf{x} \ell^i(\mathbf{x}_k)}{\|\nabla_\mathbf{x} \ell^i(\mathbf{x}_k)\|_2} \right). \tag{1}$$

Remark 1 (**Invariance of** DiBS). *For monotonic, possibly nonaffine transformations $h^1, \ldots, h^N$, let $\boldsymbol{h}(\boldsymbol{\ell})(\mathbf{x}) = [h^1(\ell^i)(\mathbf{x}), \ldots, h^N(\ell^N)(\mathbf{x})]$. Then DiBS produces the same solution for the bargaining games $\mathcal{B}_\mathcal{S}(\boldsymbol{h}(\boldsymbol{\ell}))$ and $\mathcal{B}_\mathcal{S}(\boldsymbol{\ell})$, for the same initial point $\mathbf{x}_1 \in \mathcal{S}$ and sequence of stepsizes $\{\alpha_k\}_{k\geq 0}$. This invariance to monotonic nonaffine transformations is because DiBS only utilizes normalized gradients and locally preferred states, i.e., local minima, both of which do not change under such transformations.*

We also note that prior has work has used the classical Nash bargaining solution (NBS) (Nash et al., 1950) in the context of MTL (Navon et al., 2022). However, the NBS is invariant to only affine agent (or task) loss transformations, and can change if the agent (or task) losses undergo monotonic *nonaffine* transformations.

As highlighted in Section 1, *the need for MTL methods which are robust to monotonic non-affine transformations makes it natural to consider incorporating DiBS—which is invariant to such distortions—in an MTL approach*. However, since practical MTL problems typically involve nonconvex losses, it is crucial to examine the behavior of DiBS in the nonconvex regime, a setting not yet analyzed in prior work. This serves as our motivation for the next section.

3 WHAT CAN WE SAY ABOUT DiBS IN THE NONCONVEX REGIME?

In this section we establish a convergence guarantee for DiBS in the setting when agent losses can be nonconvex, which is often the case in practical MTL problems. Currently, guarantees only exist for the case when all agents have strongly convex losses, under which DiBS enjoys global asymptotic convergence to a Pareto stationary point (Gupta et al., 2025). We begin by highlighting our assumptions.

Assumption 1. For $\mathcal{B}_{\mathcal{S}}(\boldsymbol{\ell})$, the set of Pareto stationary points lies in the interior of $\mathcal{S}$, and all $\mathbf{x}^{*,i}$ exist, are finite, and are also in the interior of $\mathcal{S}$. The agent costs ℓ^i are differentiable, and bounded below.

Assumption 2 (Relaxed in Section C). The sequence $\{\mathbf{x}_k\}_{k=1}^{\infty}$ generated by the DiBS iterations given in Equation (1) are bounded, i.e., there is a bounded set $\mathcal{D} \subseteq \mathcal{S}$ such that $\mathbf{x}_k \in \mathcal{D}\ \forall\ k \in \mathbb{N}$.

Assumption 1 is standard and ensures that the problem is well posed. Assumption 2 is a temporary assumption that we make for a clear exposition of our arguments while studying convergence of DiBS in nonconvex settings. Assumption 2 makes the DiBS iterates bounded, and we relax this assumption in Section C, showing that standard techniques from dynamical systems theory can be used to ensure DiBS iterates are bounded, forgoing the need for Assumption 2.

We now present our main theorem, the proof of which is in Appendix B.

Theorem 1. *Let $\{\mathbf{x}_k\}_{k=1}^{\infty}$ be the sequence generated by the DiBS algorithm given in Equation* (1)*, for an initial point $\mathbf{x}_1 \in \mathcal{S}$ and stepsizes that follow the Robbins-Monro conditions, i.e., $\sum_k \alpha_k = \infty$, $\sum_k \alpha_k^2 < \infty$ (Robbins & Monro, 1951). Then, under Assumptions 1 and 2, the sequence $\{\mathbf{x}_k\}_{k=1}^{\infty}$ has a subsequence that asymptotically converges to a Pareto stationary point $\mathbf{x}^{\dagger}$, i.e., $h(\mathbf{x}^{\dagger}) = 0$.*

Theorem 1 establishes that the sequence produced by the DiBS iterates has a subsequence which converges to a Pareto stationary point, even when the agent losses are nonconvex. Note that this result is similar to a guarantee presented for the Nash bargaining solution in prior work (Navon et al., 2022). However, the existing result in prior work requires the assumption that agents' loss gradients are linearly independent at all non-Pareto stationary points. It is unclear to what extent realistic MTL problems satisfy this assumption. In comparison, our result does not require such a linear independence assumption.

4 ADAPTING DiBS TO MULTITASK LEARNING

We now extend DiBS to the multitask learning setting. We begin by introducing some notation, and then describe the underlying bargaining game for MTL, which has been proposed in previous work (Navon et al., 2022).

Notation. We use $\mathbf{0}$ to denote a zero vector of appropriate dimensions, and $\mathcal{V}(\mathbf{x}, r)$ denotes a ball of radius r, centered at $\mathbf{x}$. The vector $\theta \in \mathbb{R}^n$ represents shared task parameters; any additional task-specific parameters are suppressed for notational convenience and do not influence the calculation of θ iterates in this section.

Bargaining for MTL. The goal in MTL is to train a model to perform multiple tasks. Every learning task has an associated task loss $\ell^i(\theta) : \mathbb{R}^n \to \mathbb{R}$. During training, given parameters θ, bargaining is iteratively conducted to find the next set of parameters $\theta + \Delta\theta$.

To this end, every task is represented as a distinct bargaining agent with the (minimization) objective being an approximation of the difference $\ell^i(\theta+\Delta\theta)-\ell^i(\theta)$. We consider a first-order approximation, in line with prior bargaining work for a fair comparison (Navon et al., 2022), with the i^{th} agent's objective being

$$\min_{\|\Delta\theta\|_2 \leq \epsilon} \underbrace{\left(\nabla_\theta \ell^i(\theta)\right)^\top \Delta\theta}_{:=\omega^i(\Delta\theta)},$$

where we constrain the update vector $\Delta\theta$ to lie in $\mathcal{V}(\mathbf{0}, \epsilon)$, to prevent overshooting caused by large update steps. *Thus, for an MTL problem with N tasks—the bargaining game becomes $\mathcal{B}_{\mathcal{V}(\mathbf{0},\epsilon)}([\omega^1, \ldots, \omega^N])$.*

`DiBS` for MTL. We now proceed to apply `DiBS` to the bargaining game $\mathcal{B}_{\mathcal{V}(\mathbf{0},\epsilon)}([\omega^1, \ldots, \omega^N])$. Note that the bargaining game has linear objectives, and thus the normalized gradient

$$\frac{\nabla_{\Delta\theta}\omega^i}{\|\nabla_{\Delta\theta}\omega^i\|_2} = \frac{\nabla_\theta \ell^i(\theta)}{\|\nabla_\theta \ell^i(\theta)\|_2}$$

only needs to be computed once at the starting of a bargaining game. Further, the quantity $\mathbf{x}^{*,i}$ in Equation (1) has a closed form solution $\mathbf{x}^{*,i} = -\left(\nabla_\theta \ell^i(\theta) / \|\nabla_\theta \ell^i(\theta)\|_2\right) \cdot \epsilon$ for the linear objectives, i.e., the furthest point in the negative gradient direction, given that $\Delta\theta$ is constrained to lie in $\mathcal{V}(0, \epsilon)$. Thus, for an initial choice $\Delta\theta_1$ and stepsize sequence $\{\alpha_k\}$, the `DiBS` iterates for MTL become

$$\Delta\theta_{k+1} = \Delta\theta_k - \alpha_k \cdot \sum_{i\in[N]} \left(\left\| \Delta\theta_k + \epsilon \cdot \frac{\nabla_\theta \ell^i(\theta)}{\|\nabla_\theta \ell^i(\theta)\|_2} \right\|_2 \cdot \frac{\nabla_\theta \ell^i(\theta)}{\|\nabla_\theta \ell^i(\theta)\|_2} \right), k \in \mathbb{N}$$

(`Multi-step DiBS-MTL`)

In practice, naively applying `Multi-step DiBS-MTL` can be slow. We propose the following approximation to `DiBS`, which is more computationally efficient and still *preserves* the desirable invariance to monotonic, nonaffine task loss transformations:

$$\Delta\theta = -\epsilon \cdot \sum_{i\in[N]} \frac{\nabla_\theta \ell^i(\theta)}{\|\nabla_\theta \ell^i(\theta)\|_2} \qquad \text{(DiBS-MTL)}$$

`DiBS-MTL` can be viewed as an approximation of `Multi-step DiBS-MTL`, obtained by setting the initial $\Delta\theta_1 = 0$, running the system for a single step, i.e., only for $k = 1$, and outputting $\Delta\theta_2$ as $\Delta\theta$. Such single step approximations are common for ensuring practical computational speedups in MTL methods where an iterative processes are involved, c.f. `FAMO` (Liu et al., 2023). As we will show in Section 5, `DiBS-MTL` offers competitive performance on MTL tasks, even with a single step approximation. We remark that unlike equation 1, the single step update rule `DiBS-MTL` that does not have explicit dependencies on $\mathbf{x}^{*,i}$ due to the first-order approximation combined with the ball constraint. In Section F we provide results on a demonstrative two dimensional nonconvex example showing that the single-step version performs similarly to the multi-step version despite being relatively simpler.

5 Experiments

We now evaluate `DiBS-MTL` in standard MTL benchmarks extensively used in literature (Yu et al., 2020a; Liu et al., 2021a; Navon et al., 2022; Liu et al., 2023), a demonstrative two objective example, a computer vision benchmark (`NYU-v2`) (Silberman et al., 2012), and a multitask reinforcement learning dataset (Meta-World `MT10`) (Yu et al., 2020b). Our main aims are: (i) to verify that `DiBS-MTL` reliably converges to Pareto solutions from varying initializations, (ii) to compare the overall performance of solutions identified by `DiBS-MTL` to those produced by state of the art approaches across a variety of large-scale MTL benchmarks, and (iii) to verify the robustness of `DiBS-MTL` to nonaffine task loss transformations that skew the results of other methods.

5.1 Does `DiBS-MTL` reliably converge to Pareto solutions?

Setting. We first test `DiBS-MTL` on a demonstrative two-dimensional, nonconvex multi-objective optimization example with two objectives $(\mathcal{L}_1, \mathcal{L}_2)$, which is a standard benchmark for illustrating the ability of MTL methods to achieve balanced Pareto solutions (Yu et al., 2020a; Liu et al., 2021a; Navon et al., 2022; Liu et al., 2023). The explicit forms of the objectives are provided in Appendix E.1. As shown in Figure 1, each objective in this example has deep valleys, with the bottoms having a large magnitude difference between the objective gradients, and a high (positive) curvature. Such phenomenon has been documented to exist when optimizing neural networks as well, and can lead to one objective (task) dominating others (Goodfellow et al., 2014; Yu et al., 2020a).

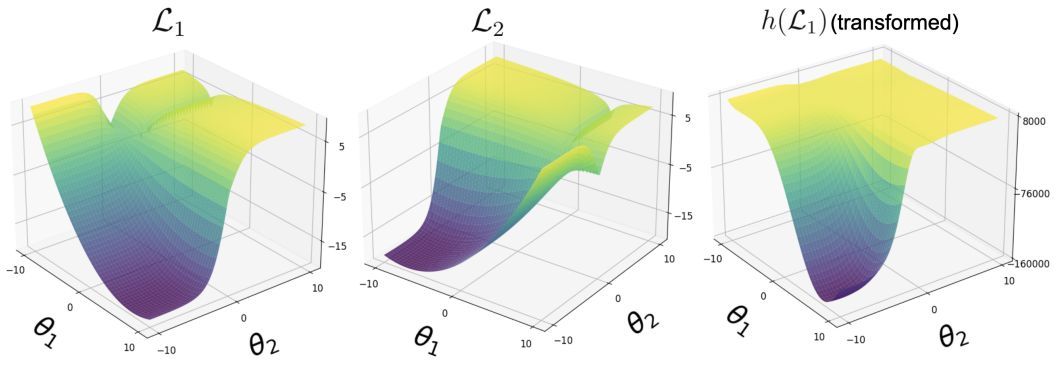


Figure 1: The loss functions used in the demonstrative two-dimensional example. The transformation we use is $h(\mathcal{L}) = \text{sign}(\mathcal{L}) \cdot \mathcal{L}^4$.

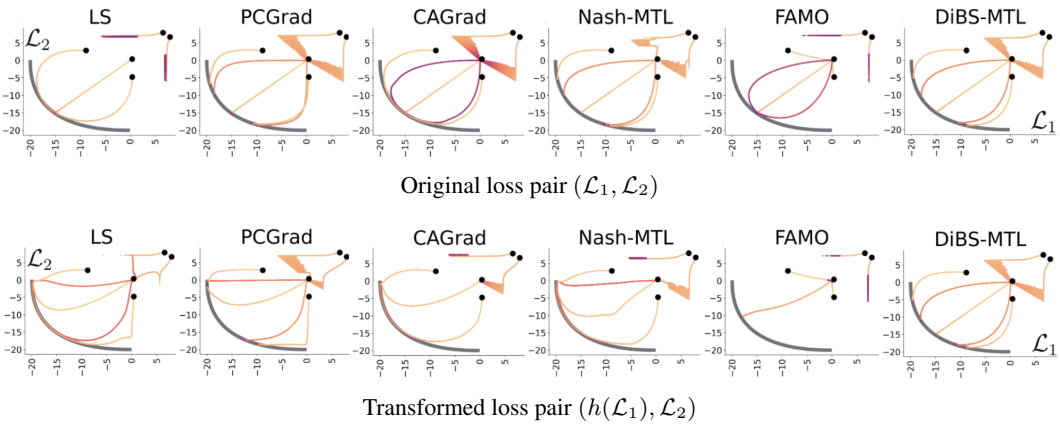


Figure 2: Results for the demonstrative two objective nonconvex problem. Each initialization is represented by •, and ▬ denotes the Pareto front. We retain the original $\mathcal{L}_1$ axis in the transformed case to better visualize the different results reached by baselines. `DiBS-MTL` provides balanced Pareto solutions invariant to monotonic nonaffine task-loss transformations, while competing methods display degraded fairness, favoring task 1.

Baselines. We test `DiBS-MTL` along with 5 baselines—(1) linear scalarization (`LS`), which optimizes the unweighted loss average, (2) `CAGrad` (Liu et al., 2021a), (3) `PCGrad` (Yu et al., 2020a), (4) `Nash-MTL` (Navon et al., 2022), and (5) `FAMO`[1] (Liu et al., 2023).

`DiBS-MTL` yields Pareto solutions from diverse initializations, which are unaffected by monotonic nonaffine transformations. Figure 2 plots the solutions generated by `DiBS-MTL` for different initializations, for the costs $(\mathcal{L}_1, \mathcal{L}_2)$ above, as well as for $(h(\mathcal{L}_1), \mathcal{L}_2)$ where we apply the monotonic nonaffine transformation $h(x) = \text{sign}(x) \cdot x^4$ to $\mathcal{L}_1$. We observe that *`DiBS-MTL` consistently leads to balanced Pareto solutions for all initializations*, and these solutions remain invariant to the transformation. It is also observed that unlike `DiBS-MTL`, all other MTL methods produce significantly biased solutions once $\mathcal{L}_1$ is transformed (i.e., comparing the top and bottom rows of Figure 2), signaling that they are susceptible to task domination when tasks are differently scaled.

5.2 How does `DiBS-MTL` compare to existing MTL methods?

To compare `DiBS-MTL` with existing methods in large-scale learning examples, we perform experiments in supervised learning (computer vision) and multitask reinforcement learning (MTRL)

[1]We were unable to reproduce the baseline plots reported in the original FAMO paper (Liu et al., 2023). The results shown here were obtained by strictly following the default parameters and instructions in the publicly available `FAMO` repository `https://github.com/Cranial-XIX/FAMO`.

Table 1: Results for supervised learning `NYU-v2` benchmark. `DiBS-MTL` is competitive across all tasks, and displays healthy overall performance, with the best average relative performance drop ($\Delta m\%$), and second-best mean rank (MR). **Bold**, underlined values indicate best, second-best performances per column respectively.

	Segmentation		Depth Estimation		Surface Normal						
	mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Median ↓	Mean ↓	< 30 ↑	< 22.5 ↑	< 11.25 ↑	$\Delta m\%$ ↓	MR ↓
STL	38.30	63.76	0.6754	0.2780	19.21	25.01	69.15	57.20	30.14		
MGDA	32.03	60.77	0.6103	0.2453	**19.00**	**24.64**	**69.83**	**57.78**	**30.55**	-0.6944	3.22
UW	39.08	64.73	0.5464	0.2285	23.04	27.34	62.85	49.23	23.49	3.7667	4.89
NashMTL	<u>40.16</u>	<u>65.65</u>	**0.5332**	**0.2204**	19.96	25.25	68.29	55.72	28.62	<u>-3.9833</u>	**2.33**
FAMO	37.58	64.08	0.5595	0.2297	<u>19.15</u>	<u>25.04</u>	<u>69.36</u>	<u>57.44</u>	<u>30.23</u>	-3.8200	3.33
LS	40.16	65.63	0.5446	0.2223	23.03	27.50	62.66	49.32	23.61	3.0569	4.22
DiBS-MTL	**40.92**	**66.60**	<u>0.5337</u>	<u>0.2217</u>	20.06	25.35	68.15	55.54	28.54	**-4.1140**	<u>2.89</u>

domains that are standard benchmarks in MTL literature (Yu et al., 2020a; Liu et al., 2021a; Navon et al., 2022; Liu et al., 2023). We begin by describing the set-up, baselines and metrics for both. We keep the default training procedures established in prior work for a fair comparison. All implementation details are given in Appendix E.

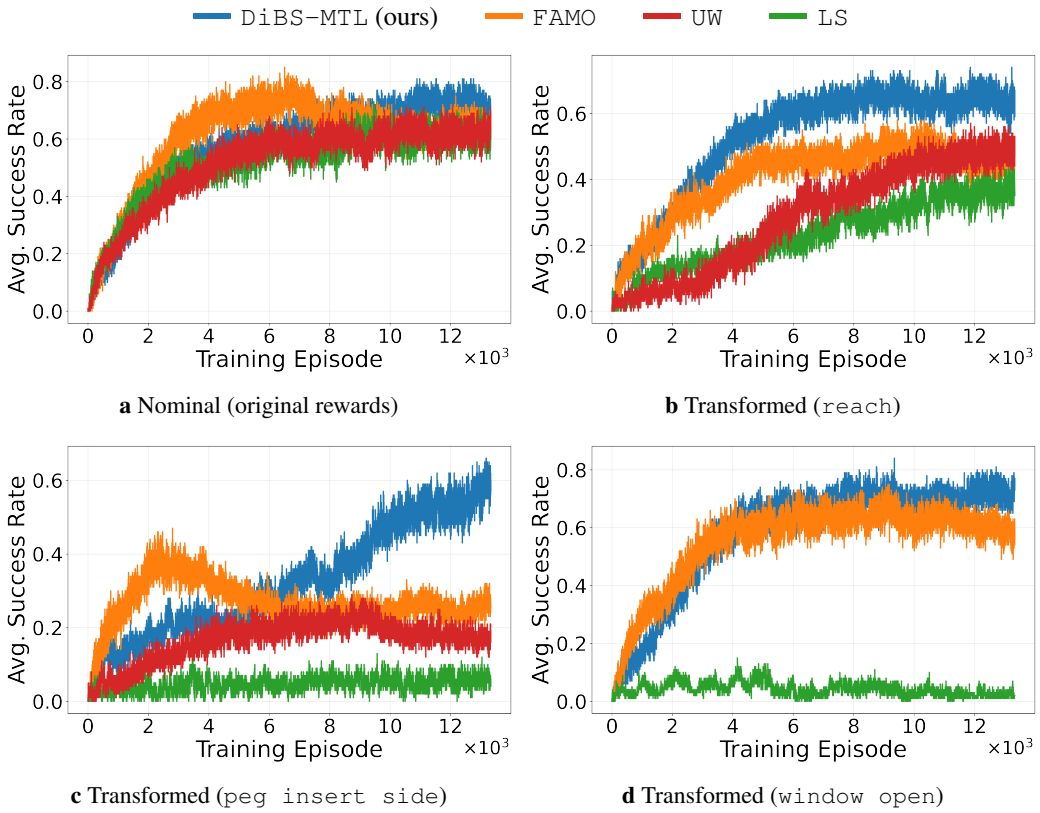
Supervised learning set-up. The setting is a supervised learning computer vision problem with 3 tasks. Specifically, we use the `NYU-v2` indoor scene dataset (Silberman et al., 2012) with 1449 RGBD images and dense per-pixel labeling for 3 learning tasks—semantic segmentation (13 classes), depth estimation, and surface normal prediction. Similar to prior work, we train a multitask attention network (MTAN) (Liu et al., 2019).

Supervised learning baselines. We use 5 baselines—(1) `LS`, (2) uncertainty weighting (`UW`) (Kendall et al., 2018), and 3 methods which have consistently outperformed other MTL approaches for this particular dataset in prior work—(3) `MGDA` (Sener & Koltun, 2018), (4) `Nash-MTL` (Navon et al., 2022), and (5) `FAMO` (Liu et al., 2023).

Supervised learning metrics. We report several task-specific metrics covering each prediction type. For semantic segmentation, we use the mean Intersection-over-Union (mIoU), which averages the per-class overlap between predicted and ground-truth regions, and the overall pixel accuracy across the image. For depth estimation, we report the mean absolute error and mean relative error. For surface normal prediction, we include the mean and median angular error between estimated and true normals, along with the percentage of pixels whose angular error falls below 30°, 22.5°, and 11.25° (Silberman et al., 2012). In addition, we report two MTL-specific metrics used in previous work—average relative per-task performance drop $\Delta m\%$, and mean rank (MR) across tasks (Navon et al., 2022; Liu et al., 2023). $\Delta m\%$ is calculated relative to `STL`—single-task learning, corresponding to learning a separate model for each task—given for a method m as $\Delta m\% = \left({}^1\!/_N \cdot \sum_{k \in [N]} {}^{(M_{m,k} - M_{\text{STL},k})}\!/_{M_{\text{STL},k}} \right) \times 100$, where $M_{i,k}$ is method i's value on metric M_k. All reported results have been averaged over 3 seeds, as done in prior work.

MTRL set-up. The setting is a multitask reinforcement learning (MTRL) problem with 10 tasks. Following prior work, we evaluate `DiBS-MTL` on the Meta-World `MT10` benchmark (Yu et al., 2020b), where a robot arm manipulator has 10 tasks, each with distinct reward functions. The complete list of tasks is given in Appendix E.

MTRL metric and baselines. We compare `DiBS-MTL` with (1) `LS`, (2) `UW`, and (3) `FAMO`. We note that results for `Nash-MTL` could not be collected due to its excessive run-time for this MTRL example (c.f., Section F.1 for a runtime comparison, where `Nash-MTL` is slow even for the simpler demonstrative example). Following prior work, for every method, we use Soft Actor-Critic (SAC) (Haarnoja et al., 2018) as the underlying reinforcement learning algorithm. Following prior work, we report the fraction of tasks successfully completed (Navon et al., 2022; Liu et al., 2023), where the definition of task success is as described in the Meta-World benchmark (Yu et al., 2020b). All reported results have been averaged over 10 random seeds. The implementation details are included in Appendix E.



a Nominal (original rewards)

b Transformed (`reach`)

c Transformed (`peg insert side`)

d Transformed (`window open`)

Figure 3: Training Curves for MTRL Meta-World `MT10` benchmark. Reward transformations significantly degrade performance of baseline MTL methods, while `DiBS-MTL` displays robustness.

`DiBS-MTL` performs competitively to state-of-the-art MTL methods. Table 1 shows the results for the 3 task supervised learning `NYU-v2` dataset. We find that `DiBS-MTL` performs consistently well for all tasks and is competitive with the baselines. Notably, `DiBS-MTL` achieves state-of-the-art performance in $\Delta m\%$ (overall average relative per-task performance drop), and second-best MR (mean rank) overall—both these metrics correspond to how well rounded an MTL method is with respect to all tasks. `DiBS-MTL` also achieves the state-of-the-art, and second-best performance in the segmentation and depth estimation tasks, respectively. In the 10 task robotic MTRL experiment as well, Table 2 and Figure 3a show that `DiBS-MTL` performs competitively, and achieves the second-best performance for the nominal case, where the reward functions for all tasks are the ones provided in Meta-World.

5.3 How robust are `DiBS-MTL` & existing MTL methods against nonaffine transformations?

We now conduct experiments in the MTRL Meta-World `MT10` benchmark Yu et al. (2020b) to investigate how `DiBS-MTL` and baseline MTL methods perform when some tasks undergo nonaffine transformations. We begin by describing the setup for the transformations. All training procedures are taken to be the same as in Section 5.2. We limit our study in this section to MTRL, because task-specific losses in computer vision are relatively standardized in the literature and thus transformations there will be unwarranted (Wang et al., 2022; Azad et al., 2023). In comparison, as we mention in Section 1, in the MTRL setting, rewards for different tasks might be designed on very different scales.

Transformed MTRL setup. We consider transformation of 3 different tasks—for each, we apply monotonic, nonaffine transformations to the underlying reward function. For actor-critic algorithms like SAC, this transformation corresponds to potentially nonmonotonic, nonaffine critic loss trans-

Table 2: Best-checkpoint success (evaluated every 200 episodes) results for MTRL Meta-World `MT10` benchmark. Results are mean $\pm$ standard error over 10 seeds. **Bold** values indicate best performances per column.

Method	Nominal	Reach	Window Open	Peg Insert
DiBS-MTL	0.890 ± 0.023	$\mathbf{0.850 \pm 0.022}$	$\mathbf{0.920 \pm 0.025}$	$\mathbf{0.700 \pm 0.037}$
FAMO	$\mathbf{0.920 \pm 0.020}$	0.700 ± 0.075	0.830 ± 0.033	0.570 ± 0.070
LS	0.850 ± 0.040	0.480 ± 0.044	0.200 ± 0.026	0.230 ± 0.030
UW	0.830 ± 0.033	0.650 ± 0.081	—[2]	0.420 ± 0.074

formations. This is because the losses fit the collected rewards to value functions rather than fitting the state dependent rewards. Thus, these task reward transformations can lead to potentially *non-monotonic* task loss transformations. However, we remark that in the policy landscape, a good policy for the nominal reward is also likely to perform well for the transformed reward.

Reward transformations. We consider the following task-transformation pairs: (i) $h(r) = \text{sign}(r) \cdot r^4$ for `reach`, (ii) $h(r) = (5 + r)^4$ for `peg insert side`, (h is monotonic over the reward range), and (iii) $h(r) = \exp(r)$ for `window open`. These transformations were arbitrarily selected to illustrate a broad range of functional forms and tasks. We use the same baselines, metric, seed values, and training procedure as the nominal MTRL experiment in Section 5.2.

`DiBS-MTL` retains performance under nonaffine reward transformations. Figures 3a to 3d and Table 2 show that `DiBS-MTL` achieves remarkable robustness to the different reward transformations compared to existing MTL methods. `DiBS-MTL` achieves state-of-the-art overall performance for all transformed cases, while other methods face significant performance degradation in one or more cases compared to their performance in the nominal setting.

6 Conclusion

We consider the problem of constructing multitask learning (MTL) methods that are robust to monotonic, nonaffine task loss transformations. In practice, different task losses can be arbitrarily scaled with respect to one another, and those scalings can be understood as monotonic, possibly nonaffine transformations of some ideal, unknown losses which are meaningfully comparable. To address this problem, we present `DiBS-MTL`, an MTL method which is invariant to such transformations because it uses only normalized task loss gradients. Building upon recent work which formalizes the MTL training update step as a bargaining game played between tasks (Navon et al., 2022), we adapt the recently introduced `DiBS` bargaining approach (Gupta et al., 2025) in order to find a Pareto stationary (and monotonic transformation-invariant) update direction. While `DiBS` enjoys convergence guarantees when losses are strongly convex, practical MTL problems almost always have nonconvex task losses. To bridge this gap, we prove that a subsequence of the `DiBS` iterates asymptotically converges to a Pareto stationary point even in the nonconvex regime. Empirically, we demonstrate that `DiBS-MTL` achieves performance competitive with state-of-the-art methods on standard large-scale MTL benchmarks. Moreover, when task losses are nonaffinely transformed other leading MTL methods suffer degraded performance, with some tasks dominating others, whereas `DiBS-MTL` still maintains high performance, demonstrating robustness to such transformations.

[2] `UW` results could not be collected for the `window open` case, because the training iterations became too unstable and the underlying MuJoCo simulator (Todorov et al., 2012) consistently gave out-of-bounds errors.

ETHICS STATEMENT

This paper presents work in the field of multitask learning. Our work has many potential applications, but we do not feel any ethical concerns need to be highlighted at this time.

REPRODUCIBILITY STATEMENT

Implementation details for all experiments are given in Section E. Additionally, we provide all of our code in the supplementary materials. We provide setup instructions in our README, including python and package versions. The theoretical proof is given in Appendix B.

REFERENCES

Reza Azad, Moein Heidary, Kadir Yilmaz, Michael Hüttemann, Sanaz Karimijafarbigloo, Yuli Wu, Anke Schmeink, and Dorit Merhof. Loss functions in the era of semantic segmentation: A survey and outlook. *arXiv preprint arXiv:2312.05391*, 2023.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33:2039–2050, 2020.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

Chris Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems*, 34:27503–27516, 2021.

Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 11543–11552, 2020.

Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *International conference on machine learning*, pp. 3854–3863. PMLR, 2020.

Kushagra Gupta, Surya Murthy, Mustafa O Karabag, Ufuk Topcu, and David Fridovich-Keil. Cooperative bargaining games without utilities: Mediated solutions from direction oracles. *arXiv preprint arXiv:2505.14817*, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Yuzheng Hu, Ruicheng Xian, Qilong Wu, Qiuling Fan, Lang Yin, and Han Zhao. Revisiting scalarization in multi-task learning: A theoretical perspective. *Advances in Neural Information Processing Systems*, 36:48510–48533, 2023.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6129–6138, 2017.

Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and Pawan K Mudigonda. In defense of the unitary scalarization for deep multi-task learning. *Advances in Neural Information Processing Systems*, 35:12169–12183, 2022.

Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021a.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. Famo: Fast adaptive multitask optimization. *Advances in Neural Information Processing Systems*, 36:57226–57243, 2023.

Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=IMPnRXEWpvr.

Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1871–1880, 2019.

Yadati Narahari. *Game theory and mechanism design*, volume 4. World Scientific, 2014.

John F Nash et al. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *International Conference on Machine Learning*, pp. 16428–16446. PMLR, 2022.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Jiayi Shen, Qi Wang, Zehao Xiao, Nanne Van Noord, and Marcel Worring. Go4align: Group optimization for multi-task alignment. *Advances in Neural Information Processing Systems*, 37: 111382–111405, 2024.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pp. 746–760. Springer, 2012.

Shagun Sodhani and Amy Zhang. Mtrl - multi task rl algorithms. Github, 2021. URL https://github.com/facebookresearch/mtrl.

Xiaozhuang Song, Shun Zheng, Wei Cao, James Yu, and Jiang Bian. Efficient and effective multi-task grouping via meta learning on task combinations. *Advances in Neural Information Processing Systems*, 35:37647–37659, 2022.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International conference on machine learning*, pp. 9120–9132. PMLR, 2020.

William Thomson. Cooperative models of bargaining. *Handbook of game theory with economic applications*, 2:1237–1284, 1994.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 9(2):187–212, 2022.

Derrick Xin, Behrooz Ghorbani, Justin Gilmer, Ankush Garg, and Orhan Firat. Do current multi-task optimization methods in deep learning even help? *Advances in neural information processing systems*, 35:13597–13609, 2022.

Rui Yu, Shenghua Wan, Yucen Wang, Chen-Xiao Gao, Le Gan, Zongzhang Zhang, and De-Chuan Zhan. Reward models in deep reinforcement learning: A survey. *arXiv preprint arXiv:2506.15421*, 2025.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020a.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020b.

A Large language model (LLM) usage disclosure

The authors used LLMs in the following ways.

- We used GPT-5 for spelling checks, grammar checks, and paraphrasing after the initial text is written by the authors.
- We used GPT-5 for help with understanding axis labeling options for plotting figures in the `matplotlib` package (Hunter, 2007).
- We used GPT-5 to resolve package dependency and python version issues that arose while installing the MuJoCo simulator (Todorov et al., 2012).

B Proof of Theorem 1

Proof sketch. As the iterates are bounded, there exists a subsequence which converges to some cluster point $\mathbf{x}^\dagger$. To show that $\mathbf{x}^\dagger$ is a Pareto stationary point, we will show by contradiction that $h(\mathbf{x}^\dagger) = 0$.

Proof. Because the `DiBS` iterates $\{\mathbf{x}_k\}_{k=1}^\infty$ is a bounded sequence from Assumption 2 (relaxed in Section C), by Bolzano-Weierstrass theorem, $\exists\ \mathbf{x}^\dagger \in \mathcal{S}$ such that a subsequence of the sequence $\{\mathbf{x}_k\}_{k=1}^\infty$ converges to $\mathbf{x}^\dagger$, and that for every neighborhood $\mathcal{U}$ of $\mathbf{x}^\dagger$, there exist infinite $n \in \mathbb{N}$ such that $\mathbf{x}_n \in \mathcal{U}$. We will show by contradiction that $h(\mathbf{x}^\dagger) = 0$. Let there exist an $a > 0$ such that $\|h(\mathbf{x}^\dagger)\|_2 = a$. We define

$$M := \sup_{\mathbf{x}\in\mathcal{D}} \|h(\mathbf{x})\|_2, \qquad u := \frac{h(\mathbf{x}^\dagger)}{\|h(\mathbf{x}^\dagger)\|_2}$$

Then, by continuity of $u^\top h(\mathbf{x})$, we have

$$\forall\ \delta > 0, \exists\ \epsilon > 0 \text{ such that } \|\mathbf{x} - \mathbf{x}^\dagger\|_2 \leq \delta \implies u^\top h(\mathbf{x}) \geq a\epsilon, \tag{2}$$

$$\text{and } \exists\ N \in \mathbb{N} \text{ such that } k \geq N \implies \alpha_k \leq \frac{a}{C} \text{ for some } C > 0. \tag{3}$$

Let $\mathcal{V}(\mathbf{x}, r)$ denote a ball in $\mathbb{R}^n$ centered at $\mathbf{x} \in \mathbb{R}^n$, with radius r. Let us pick a $\tilde{\delta} > 0$, and analyse the `DiBS` iterations when they are in $\mathcal{V}(\mathbf{x}^\dagger, \tilde{\delta})$. Because there an infinite number of n such that $\mathbf{x}_n \in \mathcal{V}(\mathbf{x}^\dagger, \tilde{\delta})$, there are two possibilities:

1. **Case 1.** $\exists\, \delta > \tilde{\delta}$ such that the DiBS iterates enter $\mathcal{V}(\mathbf{x}^\dagger, \tilde{\delta})$, exit $\mathcal{V}(\mathbf{x}^\dagger, \delta)$, and then re-enter $\mathcal{V}(\mathbf{x}^\dagger, \tilde{\delta})$—an infinite number of times.

2. **Case 2.** The DiBS iterates enter $\mathcal{V}(\mathbf{x}^\dagger, \tilde{\delta})$ and then exit it at most a finite number of times before eventually remaining in $\mathcal{V}(\mathbf{x}^\dagger, \tilde{\delta})$ forever.

Case 1. Consider the counts when the DiBS iterates are inside $\mathcal{V}(\mathbf{x}^\dagger, \tilde{\delta})$, $n < t_1 < t_2 < \dots$ such that $\mathbf{x}_{t_k} \in \mathcal{V}(\mathbf{x}^\dagger, \tilde{\delta})\ \forall\ k \in \mathbb{N}$, and let the earliest corresponding counts when the DiBS iterates are outside $\mathcal{V}(\mathbf{x}^\dagger, \delta)$ be $e_k := \min\{t \geq t_k | \mathbf{x}_t \notin \mathcal{V}(\mathbf{x}^\dagger, \delta)\}$. Then, from Equation (2), we have

$$u^\top h(\mathbf{x}_t) \geq a\epsilon\ \forall\ t = t_k, t_{k+1}, \dots, e_k - 1.$$

Now, from the definition of t_k and e_k, we have

$$\begin{aligned}
&\|\mathbf{x}_{e_k} - \mathbf{x}_{t_k}\|_2 \geq \delta - \tilde{\delta} > 0 \\
\implies &\left\| \sum_{t=t_k}^{e_k-1} (\mathbf{x}_{t+1} - \mathbf{x}_t) \right\|_2 \geq \delta - \tilde{\delta} \\
\implies &\sum_{t=t_k}^{e_k-1} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|_2 \geq \delta - \tilde{\delta} && \text{(triangle inequality)} \\
\implies &\sum_{t=t_k}^{e_k-1} \alpha_t = \sum_{t=t_k}^{e_k-1} \frac{\|\mathbf{x}_{t+1} - \mathbf{x}_t\|_2}{\|h(\mathbf{x}_t)\|_2} \geq \sum_{t=t_k}^{e_k-1} \frac{\|\mathbf{x}_{t+1} - \mathbf{x}_t\|_2}{M} \geq \frac{\left(\delta - \tilde{\delta}\right)}{M} \\
\implies u^\top (\mathbf{x}_{e_k} - \mathbf{x}_{t_k}) &= \sum_{t=t_k}^{e_k-1} u^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) = -\sum_{t=t_k}^{e_k-1} \alpha_t u^\top h(\mathbf{x}_t) \\
&\leq -a\epsilon \sum_{t=t_k}^{e_k-1} \alpha_t && \text{(from Equation (2))} \\
\implies u^\top (\mathbf{x}_{e_k} - \mathbf{x}_{t_k}) &\leq -\frac{a\epsilon \left(\delta - \tilde{\delta}\right)}{M}. && (4)
\end{aligned}$$

Here, the first equality could be legitimately written as the definition of case 1 implies that $h(\mathbf{x}_t) \neq 0, t = t_k, \dots, e_k - 1$, otherwise the iterates would have remained within $\mathcal{V}(\mathbf{x}^\dagger, \delta)$ forever. Now, from the definition of case 1, t_{k+1} and e_k, we have that the DiBS iterates cannot remain outside $\mathcal{V}(\mathbf{x}^\dagger, \tilde{\delta})$ for an infinite amount of time, and thus $\exists\, C' > 0$ such that $C' > \sum_{t=e_k}^{t_{k+1}-1} 1\ \forall\ k$. Thus

$$\begin{aligned}
u^\top \left(\mathbf{x}_{t_{k+1}} - \mathbf{x}_{e_k}\right) &= \sum_{t=e_k}^{t_{k+1}-1} u^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) \\
&\leq \sum_{t=e_k}^{t_{k+1}-1} \alpha_t M && \text{(Cauchy-Schwarz)} \\
&= M \sum_{t=e_k}^{t_{k+1}-1} \alpha_t \\
&\leq \frac{aMC'}{C} && \text{(from Equation (3))} \\
\implies u^\top \left(\mathbf{x}_{t_{k+1}} - \mathbf{x}_{e_k}\right) &\leq \frac{aMC'}{C} := a\gamma. && (5)
\end{aligned}$$

Combining Equations (4) and (5), we get

$$u^\top \left(\mathbf{x}_{t_{k+1}} - \mathbf{x}_{t_k}\right) \leq a\left(\gamma - \frac{\epsilon\left(\tilde{\delta} - \delta\right)}{M}\right)$$

$$\implies u^\top \mathbf{x}_{t_{k+1}} \leq u^\top \mathbf{x}_{t_1} + k \cdot a\left(\gamma - \frac{\epsilon\left(\tilde{\delta} - \delta\right)}{M}\right) \tag{6}$$

From Cauchy-Schwarz inequality, $\|u^\top \mathbf{x}_{t_{k+1}}\|_2 \leq \|\mathbf{x}_{t_{k+1}}\|_2$ is bounded for all $k \in \mathbb{N}$ as the DiBS sequence is bounded. However, from Equation (6), as k increases $\|u^\top \mathbf{x}_{t_{k+1}}\|_2$ becomes unbounded, which is a **contradiction**.

Case 2. By definition of case 2, $\exists\, t^* < \infty$ such that $\mathbf{x}_t \in \mathcal{V}(\mathbf{x}^\dagger, \delta)\ \forall\, t \geq t^*$. Thus for $T > t^*$, we have

$$\begin{aligned} u^\top (\mathbf{x}_T - \mathbf{x}_{t^*}) &= \sum_{t=t^*}^{T-1} u^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) \\ &= -\sum_{t=t^*}^{T-1} \alpha_t u^\top h(\mathbf{x}_t) \\ &\leq -a\epsilon \sum_{t=t^*}^{T-1} \alpha_t \qquad \text{(from Equation (2))} \\ \implies u^\top \mathbf{x}_T &\leq u^\top \mathbf{x}_{t^*} - a\epsilon \sum_{t=t^*}^{T-1} \alpha_t \end{aligned} \tag{7}$$

Similar to the argument of contradiction in case 1, $\|u^\top \mathbf{x}_{t^*}\|_2, \|u^\top \mathbf{x}_T\|_2, T \geq t^*$ are bounded. However, Equation (7) suggests that $\|u^\top \mathbf{x}_T\|$ becomes unbounded as T increases, because of the Robbins-Monro stepsize condition $\sum_k \alpha_k = \infty$, and $\sum_{k=1}^{t^*-1} \alpha_k$ is finite. Hence, we arrive at a **contradiction.**

Thus, from both cases, we get that $h(\mathbf{x}^\dagger) = 0$, and from Definition 1, $\mathbf{x}^\dagger$ is a Pareto stationary point with convex coefficients

$$\beta^i = \frac{\|\mathbf{x}^\dagger - \mathbf{x}^{*,i}\|_2 / \|\nabla_{\mathbf{x}} \ell^i(\mathbf{x}^\dagger)\|_2}{\sum_i \|\mathbf{x}^\dagger - \mathbf{x}^{*,i}\|_2 / \|\nabla_{\mathbf{x}} \ell^i(\mathbf{x}^\dagger)\|_2}.$$

□

C Relaxing Assumption 2

In the case when all agent (task) losses ℓ^i are convex, it has been established that the DiBS iterates are bounded (Gupta et al., 2025). In the non-convex setting, while the boundedness of the DiBS iterates intuitively holds for a problem with gradient conflict, i.e., task loss gradients point in opposite directions, we show that boundedness can be formally guaranteed with standard concepts from dynamical systems theory.

In particular, the following simple modification to DiBS allows us to guarantee boundedness, without changing the fact that the only fixed points of the dynamics are Pareto stationary points.

$$\mathbf{x}_{k+1} = \begin{cases} f_1(\mathbf{x}_k) := \mathbf{x}_k - h(\mathbf{x}_k), & \text{if } \|\mathbf{x}_k\|_2 \leq R \text{ (standard DiBS)} \\ f_2(\mathbf{x}_k) := \mathbf{x}_k - \alpha \frac{\mathbf{x}_k}{\|\mathbf{x}_k\|_2}, & \text{if } \|\mathbf{x}_k\|_2 \geq R + r \text{ (radially attractive)} \\ f(\mathbf{x}_k, t_k), & \text{if } R < \|\mathbf{x}_k\|_2 < R + r \text{ (convex combination)} \end{cases}$$

$$\text{where } f(\mathbf{x}_k) = (1 - g(\mathbf{x}_k))\, f_1(\mathbf{x}_k) + g(\mathbf{x}_k) f_2(\mathbf{x}_k) + z(\mathbf{x}_k, t_k),$$
$$z(\mathbf{x}_k, t_k) = \left(\frac{\|\mathbf{x}_k\|_2 - R}{r}\right) \cdot \left(1 - \frac{\|\mathbf{x}_k\|_2 - R}{r}\right) \cdot \sin(t)\boldsymbol{a}$$
$$g(\mathbf{x}) = \frac{e^{-\frac{r}{\|x\|_2 - R}}}{e^{-\frac{r}{\|x\|_2 - R}} + e^{-\frac{1}{1 - \frac{\|x\|_2 - R}{r}}}}, \text{ and } t_k = \sum_{i=1}^{k} \mathbb{1}\{R < \|\mathbf{x}_i\|_2 < R + r\}$$

Here, $\boldsymbol{a} \in \mathbb{R}^n$ is any constant vector of the same dimension as the iterates $\mathbf{x}_k$. The radius R can be taken to be some large positive number, larger that $\max_i \|\mathbf{x}^{*,i}\|_2$, and r can be any positive constant. This modification ensures that (i) near the Pareto front, the `DiBS` iterates act as usual, (ii) in case the optimization landscape is such that due to a lack of conflicting gradient nature, the iterates somehow move away from the Pareto region and outside the $\|\mathbf{x}\|_2 \leq R + r$ ball, the iterates switch to the radially attractive dynamics and return to the ball and remain bounded, (iii) the dynamics switch is smoothly carried out in between the $\|\mathbf{x}\|_2 \leq R$ and $\|\mathbf{x}\|_2 \leq R + r$ balls, and (iv) the fixed points of the dynamics do not change— the time varying term $z(\mathbf{x}_k, t_k)$ ensures that only Pareto stationary points (all inside the $\|\mathbf{x}\|_R$ ball) are the fixed points of the modified dynamics, and the dynamics do not converge to non-Pareto stationary points.

D Discussion on IMTL-G

We elaborate on our claim made in Section 2 that though `IMTL-G` (Liu et al., 2021b) can be invariant to monotonic nonaffine transformations, it can produce Pareto solutions that are heavily favor one task over the other, possibly leading to task domination issues.

`IMTL-G` is a gradient-based method, which tries to find an update direction which has an equal projection on all task gradient vectors. Though this equal projection property brings invariance to monotonic nonaffine transformations, it also renders `IMTL-G` susceptible to task domination, as illustrated by the following simple two dimensional example.

Let $\mathcal{S} = [-1, 1] \times [-1, 1]$, with two tasks $\ell^1(x, y) = x^2 + (y - 1)^2$ and $\ell^2(x, y) = x^2 + (y + 1)^2$. Then any point of the form $(0, y), y \in [-1, 1]$ is a valid solution that `IMTL-G` can give. Even though all such points are Pareto stationary solutions, only $(0, 0)$ is balanced in the sense that it is equidistant to these symmetric functions (one function is a reflection of the other with respect to the X-axis). *Thus* IMTL-G *can output a point like* $(0, 0.9)$ *which is heavily biased towards task 1.* In contrast, even if the iteratations are started at $(0, 0.9)$, `DiBS` will return the balanced point $(0, 0)$ as a solution.

E Experimental Details and Code base

E.1 Demonstrative Example

Loss Functions. The non-convex objectives used in $(\mathcal{L}_1, \mathcal{L}_2)$ are given by

$$\mathcal{L}_1(\theta) = c_1(\theta) f_1(\theta) + c_2(\theta) g_1(\theta), \quad \mathcal{L}_2(\theta) = c_1(\theta) f_2(\theta) + c_2(\theta) g_2(\theta)$$
$$c_1(\theta) = \max\big(\tanh(0.5\,\theta_2), 0\big), \quad c_2(\theta) = \max\big(\tanh(-0.5\,\theta_2), 0\big),$$
$$f_1(\theta) = \log\Big(\max\big(|0.5(-\theta_1 - 7) - \tanh(-\theta_2)|, 10^{-6}\big)\Big) + 6,$$
$$f_2(\theta) = \log\Big(\max\big(|0.5(-\theta_1 + 3) - \tanh(-\theta_2) + 2|, 10^{-6}\big)\Big) + 6,$$
$$g_1(\theta) = \frac{(-\theta_1 + 7)^2 + 0.1(-\theta_2 - 8)^2}{10} - 20, \quad g_2(\theta) = \frac{(-\theta_1 - 7)^2 + 0.1(-\theta_2 - 8)^2}{10} - 20.$$

As mentioned in Section 5, the transformation used on $\mathcal{L}_1$ for the transformed case is the monotonic, nonaffine transformation $h(\ell) = \text{sign}(\ell) \cdot \ell^4$.

E.2 NYUV2

Benchmark and Setup. The NYU-v2 indoor scene dataset Silberman et al. (2012) provides 1,449 RGB-D images with dense per-pixel annotations for three tasks: semantic segmentation (13 classes),

Table 3: Hyperparameters used for MT10 (v1) SAC.

Component	Value
Encoder feature dim	50
Discount γ	0.99
Initial temperature α_0	1.0
Actor update freq.	1 step / env step
Critic target τ	0.005
Target update freq.	1 step / env step
Encoder EMA τ_{enc}	0.05
Learning Rate	0.025
Update-weights cadence	every step ($= 1$)

depth estimation, and surface normal prediction. Training uses standard task losses: per-pixel cross-entropy for segmentation, masked L1 for depth estimation over valid pixels, and a masked cosine loss over valid pixels for surface normal prediction. These choices match the NYU-v2 multitask protocol followed by prior multitask learning work (Liu et al., 2023; Navon et al., 2022). Our implementation builds on the official Nash-MTL code base[3] and incorporates the FAMO implementation[4].

Model. The model architecture used in our experiments is a Multitask Attention Network (MTAN) (Liu et al., 2019) built on top of SegNet (Badrinarayanan et al., 2017). The network takes an RGB image as input and produces three task-specific outputs via separate heads: (i) a per-pixel class-score map for semantic segmentation, (ii) a scalar depth map for depth estimation, and (iii) a 3-channel surface-normal map for surface-normal prediction. Following the experimental setup of Nash-MTL (Navon et al., 2022), all multitask learning methods train this shared architecture using their respective update rules.

Training Protocol. For training, we follow the procedure outlined in (Navon et al., 2022). We train for 200 epochs with Adam (Kingma & Ba, 2014), using an initial learning rate of 1×10^{-4} reduced to 5×10^{-5} after 100 epochs. All reported results are averaged over three random seeds, namely $1, 7$, and 42.

E.3 META-WORLD MT10

Benchmark and Setup. The MetaWorld MT10 (v1) benchmark comprises ten robotic manipulation tasks: `Reach`, `Push`, `Pick-and-Place`, `Door Open`, `Drawer Open`, `Drawer Close`, `Button Press (Top-Down)`, `Peg Insert (Side)`, `Window Open`, and `Window Close` (Yu et al., 2020b). Each task has distinct reward functions and success criteria. We build on the MTRL code base (Sodhani & Zhang, 2021)[5] with MetaWorld[6].

Policy and Training Protocol. We train a single Soft Actor–Critic (SAC) policy shared across all ten tasks. Multitask learning (MTL) methods are applied to balance the actor and critic updates within the shared SAC. For each MTL method, we train for $2{,}000{,}000$ environment steps in total. Episodes have length 150; this corresponds to $\approx 13{,}333$ episodes overall. Evaluation is performed every 200 episodes ($30{,}000$ steps) on all tasks, and reported metrics are averaged over 10 random seeds: $1, 2, 3, 4, 5, 6, 7, 8, 9, 10$.

Hyperparameters. Key hyperparameter values are provided in Table 3. We follow the same hyperparemeters defined in prior works (Navon et al., 2022; Liu et al., 2023).

Reproducibility Note. During preliminary experiments, we observed that using the same random seed did not always produce identical training curves. This discrepancy arose because the original

[3] `https://github.com/AvivNavon/nash-mtl`
[4] `https://github.com/Cranial-XIX/FAMO`
[5] `https://github.com/facebookresearch/mtrl`
[6] `https://github.com/Farama-Foundation/Metaworld`

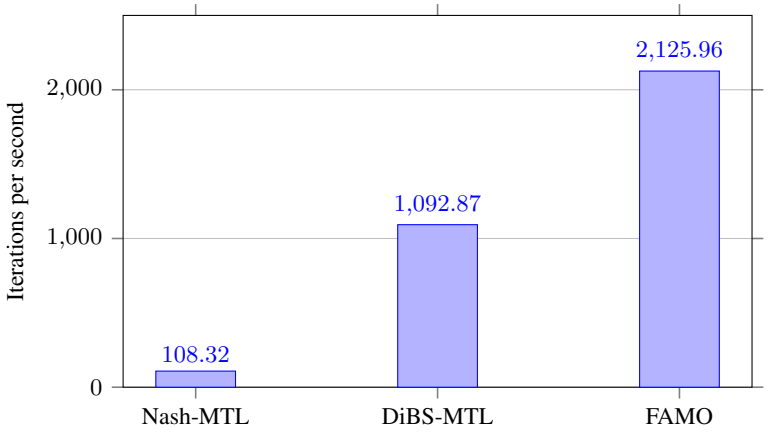


Figure 4: Average iterations per second in the non-transformed setting

MTRL code (Sodhani & Zhang, 2021) did not seed the initial action sampling during the exploration phase. As a result, small differences in early exploratory actions (first 10 steps) propagated through training and produced slight variation in learning behavior even with fixed seeds.

F Additional Results

F.1 Analysis of DiBS-MTL Runtime

In addition to examining the solutions obtained in the demonstrative example, we also evaluated the computational speed of the different MTL methods. Specifically, we compared DiBS-MTL with Nash-MTL and FAMO. We measured the number of iterations processed per second, with higher values indicating faster performance.

In Figure 4, we compare the runtimes of DiBS-MTL , FAMO, and Nash-MTL. DiBS-MTL runs substantially faster than Nash-MTL, achieving roughly half the speed of FAMO. This behavior is expected: unlike Nash-MTL, DiBS-MTL does not solve a separate optimization problem at each iteration, but it does recompute gradients and preferred states, which increases runtime relative to FAMO, which is a loss-based method and is expected to be faster.

F.2 Additional Results in the demonstrative Example

In addition to the results reported in section Section 5.1, we also ran additional methods in the demonstrative two-objective example. Specifically, we ran the Multiple Gradient Descent Algorithm (MGDA) (Désidéri, 2012), Uncertainty Weighting (UW) (Kendall et al., 2018), Impartial Multitask Learning (IMTL-G) (Liu et al., 2021b), and Multi-step DiBS-MTL. We run Multi-step DiBS-MTL for 10 steps per update. The loss functions and number of steps are identical to the experiment described in Section 5.1.

In Figure 5, we observe that the additional baseline methods of MGDA, UW, IMTL-G are not invariant to the transform, with UW failing to reach the Pareto-front in both cases. We note that Multi-step DiBS-MTL performs similarly to single step DiBS-MTL . Multi-step DiBS-MTL reaches the same points on the Pareto-front in both cases, showing it is also invariant to monotone non-affine transforms.

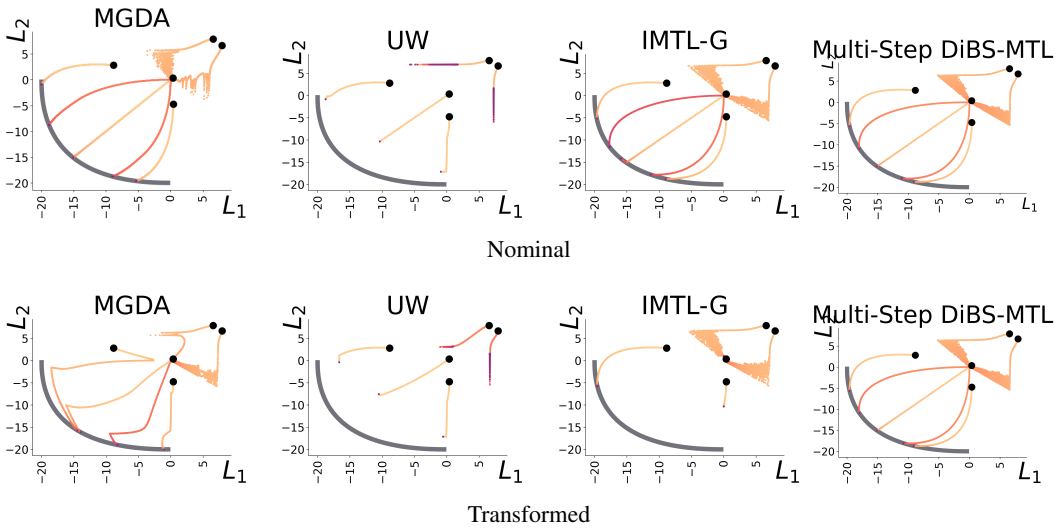


Figure 5: Additional experiments performed in the demonstrative two-objective example. We observe that, `Multi-step DiBS-MTL` exhibits the same invariance to monotone non-affine transforms as single-step `DiBS-MTL`, and also demonstrates similar behaviour.