Variational Multi-Objective Coordination

Diederik M. Roijers
Univeristy of Amsterdam
The Netherlands
d.m.roijers@uva.nl

Shimon Whiteson
University of Oxford
United Kingdom
shimon.whiteson@cs.ox.ac.uk

Alex Ihler
UC Irvine
United States
ihler@ics.uci.edu

Frans A. Oliehoek
University of Liverpool (UK)
University of Amsterdam (NL)
frans.oliehoek@liverpool.ac.uk

Abstract

In this paper, we propose *variational optimistic linear support (VOLS)*, a novel algorithm that finds bounded approximate solutions for *multi-objective coordination graphs (MO-CoGs)*. VOLS builds and improves upon an existing exact algorithm called *variable elimination linear support (VELS)*. Like VELS, VOLS solves a MO-CoG as a series of scalarized single-objective coordination graphs. We improve upon VELS in two important ways. Firstly, where VELS uses a single-objective solver called *variable elimination (VE)* as a subroutine, VOLS uses a variational method called *weighted mini-buckets (WMB)*. Because variational methods scale much better than VE, VOLS can be used to solve much larger MO-CoGs than was previously possible. Furthermore, we show that because WMB computes bounded approximations, so does VOLS. Secondly, we leverage the insight that VOLS can hot-start each call to WMB by reusing the *reparameterizations* output by WMB on earlier calls. We show empirically that VOLS scales much better than VELS and introduces only negligle error. Our experimental results indicate that the reuse of reparameterizations keeps the runtime low and the approximation quality high.

1 Introduction

In cooperative multi-agent decision problems, agents must coordinate their actions to maximize their common utility. Doing so efficiently requires exploiting *loose couplings*: each agent's behavior directly affects only a subset of the other agents. Such independence can be captured in a graphical model called a *coordination graph* [3, 5]. Typically, the common utility is codified as a sum over local scalar payoffs. However, many real-world decision problems have multiple (possibly conflicting) objectives, in which case the team utility is more naturally expressed with vector-valued payoffs. If the relative importance of these objectives is not known when the problem needs to be solved, multi-objective methods are needed to compute the set of all possibly optimal solutions [8].

In this paper, we consider the highly prevalent setting in which the true scalar value of any solution is a linear combination of the value in each objective, though the weights of this combination are unknown. In this case, the set of possibily optimal solutions is the *convex coverage set* (CCS), a subset of the *Pareto front*.[1] The CCS is typically much easier to compute than the Pareto front and is thus the solution concept of choice when it is applicable.

A state-of-the-art approach to computing the CCS for any multi-objective decision problem is the *optimistic linear support (OLS)* framework [10], which incrementally constructs the CCS by solving a series of single-objective problems. OLS is not only generic, as it can be applied to any problem for which a single-objective solver is available, but also fast, outperforming alternative approaches for small and medium numbers of objectives.

[1] If stochastic solutions are allowed, then the CCS suffices even if objectives are combined nonlinearly [13].

In this paper, we consider how OLS can best be applied to multi-objective coordination graphs. In particular, we build off an existing approach called *variable elimination linear support (VELS)*, which uses *variable elimination (VE)* [5, 11] as its single-objective solver. Since VE is an exact method, VELS produces exact CCSs. However, VE's runtime is exponential in the coordination graph's induced width, limiting scalability. Furthermore, the latest insights in OLS, i.e., that the results of calls to single-objective earlier in the series can be reused to hot-start calls later in the series [9], do not apply to VELS.

Fortunately, OLS does not require an exact single-objective solver like VE. On the contrary, given a bounded approximate single-objective solver, it computes a bounded approximation of the CCS [7]. Therefore, we propose a new approach called *variational optimistic linear support (VOLS)*, which improves upon VELS in two ways. First, it uses a variational method called *weighted mini-buckets (WMB)* [6], as the single-objective solver. Because variational methods scale much better than VE, VOLS achieves unprecedented scalability. In addition, since WMB computes bounded approximations, VOLS does so too. Second, we leverage the key insight that VOLS can hot-start each call to WMB by reusing the *reparameterizations* output by WMB on earlier calls. Our experimental results indicate that VOLS scales much better than VELS and introduces only negligle error into the resulting CCSs. Furthermore, we show that the reuse of reparameterizations improves both the runtime low and the approximation quality.

2 Background

We start with background on multi-objective decision problems, OLS, and WMB.

2.1 Multi-Objective Decision Problems

In multi-objective decision problems, there are d objectives and a vector-valued payoff function. As a result, each solution, $\mathbf{a}$, (e.g., a joint action of the agents in a coordination problem) has a vector-valued payoff $\mathbf{u}(\mathbf{a})$ of length d. In such settings, there can be multiple solutions whose value vectors are optimal for different preferences over the objectives. Such preferences can be expressed using a scalarization function $f(\mathbf{u}(\mathbf{a}), \mathbf{w})$ that is parameterized by a parameter vector $\mathbf{w}$ and returns $u_{\mathbf{w}}(\mathbf{a})$, the *scalarized* payoff of $\mathbf{a}$. When $\mathbf{w}$ is known beforehand, it is possible to *a priori* scalarize the decision problem and apply standard single-objective solvers. However, when $\mathbf{w}$ is unknown when the problem needs to be solved, we need an algorithm that computes a set of solutions containing at least one solution with maximal scalarized payoff for *each possible* $\mathbf{w}$.

In many real-world problems, f is linear, i.e., $u_{\mathbf{w}}(\mathbf{a}) = f(\mathbf{u}(\mathbf{a}), \mathbf{w}) = \mathbf{w} \cdot \mathbf{u}(\mathbf{a})$, where $\mathbf{w}$ is a vector of non-negative weights that sum to 1. In this case, a sufficient solution set is the *convex hull (CH)*, the set of all payoff vectors of undominated solutions under a linear scalarization:

$$CH(A) = \{\mathbf{u}(\mathbf{a}) : \mathbf{a} \in A \ \wedge \ \exists \mathbf{w} \forall (\mathbf{a}' \in A) \ \mathbf{w} \cdot \mathbf{u}(\mathbf{a}) \geq \mathbf{w} \cdot \mathbf{u}(\mathbf{a}')\},$$

where A is the solution space. However, the entire CH may not be necessary. Instead, it also suffices to compute a *convex coverage set (CCS)*, a lossless subset of the CH. For each possible $\mathbf{w}$, a CCS contains at least one payoff vector from the CH that has the maximal scalarized value for $\mathbf{w}$.

If f is nonlinear, we might require the *Pareto front (PF)*, a superset of the CH. However, when *stochastic solutions* are allowed, all values on the PF can be constructed by randomizing over CCS solutions [13]. Therefore, the CCS is inadequate only if the scalarization function is nonlinear *and* stochasticity is forbidden. For simplicity, we assume linear scalarizations in this paper, but our methods are also applicable to nonlinear scalarizations as long as stochastic solutions are allowed. Using the CCS, we can define a *scalarized value function*:

$$u^*_{CCS}(\mathbf{w}) = \max_{\mathbf{u}(\mathbf{a}) \in CCS} \mathbf{w} \cdot \mathbf{u}(\mathbf{a}),$$

which returns, for each $\mathbf{w}$, the maximal scalarized value achievable for that weight. $u^*_{CCS}(\mathbf{w})$ is a *piecewise-linear and convex (PWLC)* function over weight space, a property that can be exploited to construct a CCS efficiently. When the CCS cannot be computed exactly, we can often use an alternative set of payoff vectors X that approximates the CCS. The approximate scalarized value function using X,

$$u^*_X(\mathbf{w}) = \max_{\mathbf{u}(\mathbf{a}) \in X} \mathbf{w} \cdot \mathbf{u}(\mathbf{a}),$$

is also PWLC. A set X is called an ε-CCS when the maximum scalarized error across all weights is at most ε:

$$\forall \mathbf{w}, u^*_{CCS}(\mathbf{w}) - u^*_X(\mathbf{w}) \leq \varepsilon.$$

2.2 Optimistic Linear Support

Optimistic linear support (OLS) [10] solves a series of linearly scalarized instances of the multi-objective problem. The solution $\mathbf{a}^*$ to an instance scalarized with $\mathbf{w}$ maximizes $u_{\mathbf{w}}(\mathbf{a}^*) = \mathbf{w} \cdot \mathbf{u}(\mathbf{a}^*)$. When $\mathbf{a}^*$ is identified, $\mathbf{u}(\mathbf{a}^*)$ is added to a set X, which eventually becomes a CCS.

In order to select good $\mathbf{w}$'s for scalarization, OLS exploits the observation that $u^*_X(\mathbf{w})$ is PWLC over the weight simplex. In particular, OLS selects only so-called *corner weights* that lie at the intersections of line segments of the PWLC function $u^*_X(\mathbf{w})$ that correspond to the value vectors found so far. For example, in Figure 1 (left) there are two payoff vectors in X, and there is one corner weight. When OLS scalarizes the MO-CoG at this corner weight and solves it using a single-objective solver, it finds a new payoff vector, as shown in Figure 1 (right), improving it at the corner weight (as indicated by the red dashed line).

OLS prioritizes corner weights according to an optimistic estimate of their potential error reduction. The maximal potential error reduction that can be made by identifying a new payoff vector $\mathbf{u}(\mathbf{a})$ is guaranteed to be at one of these corner weights [2]. In Figure 1, the potential error reduction is denoted with dashed blue vertical lines. Note that the figure assumes that the single-objective solver is exact.

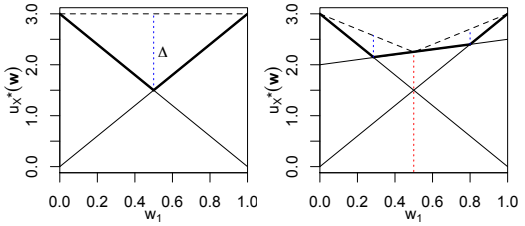

Figure 1: *(Left)* The scalarized value as a function of weights $u^*_X(\mathbf{w})$ (bold segments) for $X = \{(0,3),(3,0)\}$. There is one corner weight: $(0.5, 0.5)$. *(Right)* Adding a new payoff vector, $(2.0, 2.5)$, to X, thereby improving $u^*_X(\mathbf{w})$.

If this assumption holds, OLS is guaranteed to produce an exact CCS after solving a finite number of single-objective problems. If the single-objective solver returns a bounded approximation instead, OLS inherits this quality bound.

Lemma 1. *When an approximate single-objective solver produces a bounded approximate solution for each scalarized problem, with an error bound of at most ε, OLS produces an ε-CCS [7].*

2.3 Multi-Objective Coordination Graphs

A *multi-objective coordination graph* (MO-CoG) [10] is a multi-objective decision problem that can be formally defined as a tuple $\langle \mathcal{D}, \mathcal{A}, \mathcal{U} \rangle$ where $\mathcal{D} = \{1, ..., n\}$ is the set of n agents; $\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_n$ is the set of all possible joint actions $\mathbf{a}$, the Cartesian product of the finite action spaces of all agents; and $\mathcal{U} = \left\{\mathbf{u}^1, ..., \mathbf{u}^\rho\right\}$ is the set of ρ, d-dimensional *local payoff functions*. A local payoff function has limited scope e, i.e., only a subset of agents participate in it. The total team payoff is the (vector) sum of all local payoffs: $\mathbf{u}(\mathbf{a}) = \sum_{\mathbf{u}^e \in \mathcal{U}} \mathbf{u}^e(\mathbf{a}_e)$. We refer to the set of all possible payoff vectors as $\mathcal{V} = \{\mathbf{u}(\mathbf{a}) : \mathbf{a} \in \mathcal{A}\}$. For convenience, we assume that $\mathcal{V}$ contains both the values and associated joint actions. The additive function $\mathbf{u}(\mathbf{a})$ can be expressed as a graphical model, i.e., a factor graph [1], where the agents are the vertices and the local payoff functions are the hyperedges connecting these vertices.

A MO-CoG can be linearly scalarized with a weight vector $\mathbf{w}$. Because of the additive nature of $\mathbf{u}(\mathbf{a})$, this scalarization distributes over the local payoff functions: $u_{\mathbf{w}}(\mathbf{a}) = \sum_{\mathbf{u}^e \in \mathcal{U}} \mathbf{w} \cdot \mathbf{u}^e(\mathbf{a}_e)$. A *scalarized MO-CoG* is thus a single objective problem where the set of vector-valued local payoffs is replaced by a set of local payoff functions scalarized with a weight $\mathbf{w}$:

$$\mathcal{U}_{\mathbf{w}} = \{u^e_{\mathbf{w}}(\mathbf{a}_e) = \mathbf{w} \cdot \mathbf{u}^e(\mathbf{a}_e) : \mathbf{u}^e(\mathbf{a}_e) \in \mathcal{U}\}.$$

For convenience, we use just $\mathcal{U}_{\mathbf{w}}$ to refer to a scalarized MO-CoG as $\mathcal{D}$ and $\mathcal{A}$ are the same as in the MO-CoG. Because a scalarized MO-CoG is a single objective coordination graph, the maximal payoff can be determined with single-objective methods such as variable elimination, or, as we do in this paper, with *variational methods*.

2.4 Variational Methods for Graphical Models

Variational techniques [12, 14] can be used to bound the maximal payoff of a single-objective coordination graph. The dual decomposition approach relaxes the combinatorial optimization into an easily evaluated bound, $\max_{\mathbf{a}} u(\mathbf{a}) = \max_{\mathbf{a}} \sum_{u^e \in \mathcal{U}} u^e(\mathbf{a}_e) \leq \sum_{\mathbf{u}^e \in \mathcal{U}} \max_{\mathbf{a}_e} u^e(\mathbf{a}^e) = \bar{u}$. Then, this bound is iteratively tightened by re-parameterizing the individual functions u^e, i.e.,

one finds a set of equivalent local payoffs u'^e such that the total payoff is unchanged, $u'(\mathbf{a}) = \max_{\mathbf{a}} \sum_{\mathbf{u}'^e \in \mathcal{U}'} u'^e(\mathbf{a}_e) = u(\mathbf{a}_e)$, while minimizing the decomposed upper bound. The resulting optimization is convex and can be solved using a number of gradient-based or fixed-point techniques [12]; our implementation uses a fixed-point update based on *weighted mini-bucket* [6].

The upper bound corresponds to an optimization of the individual $u^e(\mathbf{a}_e)$; if the optimal local actions $\mathbf{a}_e^*$ are all consistent with some $\mathbf{a}^*$, then $\mathbf{a}^*$ is also the global optimum of $u(\mathbf{a})$. In practice, the decomposition bound may not be able to find the global optimum; for this reason, they typically also assemble a joint action $\mathbf{a}_l$ using, for example, greedy assignment. For each local payoff $u'^e(\mathbf{a}_e)$, we assign the elements of $\mathbf{a}_e$ that are not already assigned in $\mathbf{a}_l$ by maximizing the local function, conditioned on the already-assigned elements. The joint action $\mathbf{a}_l$ then provides a lower bound on the optimal payoff, $\underline{u} = u(\mathbf{a}_l)$. We use this upper bound $\bar{u}$, and the lower bound action $\mathbf{a}_l$, to produce a bounded approximate CCS, in accordance with Lemma 1, in our new algorithm, described below.

3 Variational Optimistic Linear Support

In this section, we present our main contribution, *variational optimistic linear support (VOLS)*. To our knowledge, VOLS is the first variational algorithm for solving MO-CoGs. Like previous OLS-based algorithms VOLS solves a MO-CoG as a series of single-objective coordination graphs. However, instead of relying on an exact single-objective solver, VOLS uses a variational method and can thus find approximate CCSs for much larger MO-CoGs than previous methods.

VOLS uses a variational subroutine for solving scalarized instances of the MO-CoG. This subroutine takes a scalarized MO-CoG as input. As output, the subroutine procudes a lower-bound joint action $\mathbf{a}_l$, which we use to construct the approximate CCS. It also produces an upper bound $\bar{u}$ on the optimal value, which we use to bound the quality of the final approximate CCS, and to prioritize instances in the series of single-objective problems to solve. Furthermore, the variational method manipulates the set of scalarized local payoff functions $\mathcal{U}_{\mathbf{w}}$ to output a reparameterization, i.e., a set of manipulated local payoff functions $\mathcal{U}'_{\mathbf{w}}$ for which all joint actions have the same (scalar) payoff, i.e., $\forall \mathbf{a} \sum_{u^e \in \mathcal{U}_{\mathbf{w}}} u^e(\mathbf{a}_e) = \sum_{u^g \in \mathcal{U}'_{\mathbf{w}}} u^g(\mathbf{a}_g)$. A key insight is that we can re-use $\mathcal{U}'_{\mathbf{w}}$ to hot-start the reparameterization of a new scalarized instance for a new weight vector $\mathbf{z}$ close to $\mathbf{w}$. Specifically, if we define the *difference graph* between two scalarization weights $\mathbf{w}$ and $\mathbf{z}$ as $\mathcal{U}_{\mathbf{w}\to\mathbf{z}} = \{u^e_{\mathbf{w}\to\mathbf{z}}(\mathbf{a}_e) = (\mathbf{z}-\mathbf{w}) \cdot \mathbf{u}^e(\mathbf{a}_e) : \mathbf{u}^e(\mathbf{a}_e) \in \mathcal{U}\}$, then adding this difference graph to the reparameterization $\mathcal{U}'_{\mathbf{w}}$ yields a valid reparameterization for $\mathbf{z}$, $\hat{\mathcal{U}}_{\mathbf{z}} = \mathcal{U}'_{\mathbf{w}} \cup \mathcal{U}_{\mathbf{w}\to\mathbf{z}}$. When $\mathbf{w}$ is close to $\mathbf{z}$, the magnitude of the local payoff functions in $\mathcal{U}_{\mathbf{w}\to\mathbf{z}}$ is small, and $\hat{\mathcal{U}}_{\mathbf{z}}$ is close to $\mathcal{U}'_{\mathbf{w}}$. Intuitively, $\hat{\mathcal{U}}_{\mathbf{z}}$ is therefore likely to be closer to the eventual reparameterization $\mathcal{U}'_{\mathbf{z}}$ that the variational subroutine will produce for $\mathcal{U}_{\mathbf{z}}$, than $\mathcal{U}_{\mathbf{z}}$ itself would be, and fewer iterations of the variational method will be required to further tighten the bounds and find $\mathcal{U}'_{\mathbf{z}}$.

The variational optimistic linear support (VOLS) algorithm (presented in Algorithm 1) takes a MO-CoG $\langle \mathcal{D}, \mathcal{A}, \mathcal{U} \rangle$ and a variational single-objective coordination graph subroutine, `variationalSOSolver`, as input. Following the OLS framework, VOLS keeps a set X, that will become an approximate CCS (line 1), and a set of upper bounds on the optimal values that VOLS finds for scalarized instances (for individual $\mathbf{w}$), U_{old} (line 2). The algorithm starts looking for solutions (i.e., approximately optimal joint actions and payoffs) for the extrema of the weight simplex (line 3–4). VOLS keeps a set $\mathcal{R}$ (line 5) with tuples of weights $\mathbf{w}$ and reparameterizations produced at those $\mathbf{w}$ by `variationalSOSolver` in iterations of the main loop.

In the main loop (lines 6–16), VOLS iteratively pops a corner weight $\mathbf{w}$ off the priority queue Q and solves the corresponding scalarized MO-CoG, $\mathcal{U}_{\mathbf{w}}$. However, instead of just calling the single-objective solver for $\mathcal{U}_{\mathbf{w}}$ directly, VOLS first looks for the reparameterization $\mathcal{U}'_{\mathbf{v}}$ found in earlier iterations (on line 10), for the weight closest to $\mathbf{w}$. Because $\forall \mathbf{a} : \sum_{u^e \in \mathcal{U}_{\mathbf{v}}} u^e(\mathbf{a}_e) = \sum_{u^g \in \mathcal{U}'_{\mathbf{v}}} u^g(\mathbf{a}_g)$, adding the difference graph $\mathcal{U}_{\mathbf{v}\to\mathbf{w}} = \{u^e_{\mathbf{v}\to\mathbf{w}}(\mathbf{a}_e) = (\mathbf{w}-\mathbf{v}) \cdot \mathbf{u}^e(\mathbf{a}_e) : \mathbf{u}^e(\mathbf{a}_e) \in \mathcal{U}\}$, results in a graph $\hat{\mathcal{U}}_{\mathbf{w}} = \mathcal{U}'_v \cup \mathcal{U}_{\mathbf{v}\to\mathbf{w}}$, for which $\forall \mathbf{a} \sum_{u^e \in \hat{\mathcal{U}}_{\mathbf{w}}} u^e(\mathbf{a}_e) = \sum_{u^g \in \mathcal{U}_{\mathbf{w}}} u^g(\mathbf{a}_g)$. In other words, reusing the reparameterization for $\mathbf{v}$ on the scalarized graph for $\mathbf{w}$ does not affect the scalarized payoff $u_{\mathbf{w}}(\mathbf{a})$ for any $\mathbf{a}$.

Besides the reparameterization $\mathcal{U}'_v \cup \mathcal{U}_{\mathbf{v}\to\mathbf{w}}$, `variationalSolver` is also provided with the joint action $\mathbf{a}_{\mathbf{v}}$ that achieves the lower bound of the previous weight $\mathbf{v}$. This joint action can be reused as an initial *guess* for the joint action at $\mathbf{w}$. If at any time during the execution of `variationalSolver` for $\mathcal{U}'_v \cup \mathcal{U}_{\mathbf{v}\to\mathbf{w}}$, the upper bound is achieved by $\mathbf{a}_{\mathbf{v}}$, the variational solver can stop. Such lower bound

Algorithm 1: VOLS($\langle \mathcal{D}, \mathcal{A}, \mathcal{U} \rangle$, variationalSOSolver)

1 $X \leftarrow \emptyset$; // approximate CCS of multi-objective payoff vectors $\mathbf{u}(\mathbf{a})$
2 $U_{old} \leftarrow \emptyset$; // set of previous $\mathbf{w}$ and $\bar{u}_{\mathbf{w}}$, for determining optimistic estimates for new corner weights
3 $Q \leftarrow$ an empty priority queue ; // a priority queue with corner weights to search
4 Add extrema of the weight simplex to Q with infinite priority;
5 $\mathcal{R} \leftarrow \emptyset$; // set of reparameterizations, joint actions, and associated weights
6 **while** $\neg Q$.isEmpty() $\wedge \neg$*timeOut* **do**
7 $\quad \mathbf{w} \leftarrow Q$.dequeue(); // retrieve a weight vector
8 $\quad \mathcal{U}'_{\mathbf{v}}, \mathbf{a}_{\mathbf{v}} \leftarrow$ select previous reparameterization and joint action found for the closest weight $\mathbf{v}$ to $\mathbf{w}$ from $\mathcal{R}$;
9 $\quad \mathcal{U}'_{\mathbf{w}}, \mathbf{a}_l, \bar{u}_{\mathbf{w}} \leftarrow$ variationalSOSolver$(\mathcal{U}'_{\mathbf{v}} \cup \mathcal{U}_{\mathbf{v}\to\mathbf{w}})$; // a variational single objective solver.
10 $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{(\mathbf{w}, \mathcal{U}'_{\mathbf{w}}, \mathbf{a}_l)\}$; // store the reparameterization of the scalarized graph for reuse
11 $\quad U_{old} \leftarrow U_{old} \cup \{(\mathbf{w}, \bar{u}_{\mathbf{w}})\}$; // store upper bound for $\mathbf{w}$, for determining the next max. possible improv.
12 $\quad$ **if** $\mathbf{u}(\mathbf{a}_l) \notin X$ **then**
13 $\quad\quad X \leftarrow X \cup \{\mathbf{u}(\mathbf{a}_l)\}$; // add lower bound payoff and associated action, $\mathbf{u}(\mathbf{a}_l)$, to the approximate CCS
14 $\quad\quad W \leftarrow$ compute new corner weights and max. possible improvements $(\mathbf{w}, \Delta_{\mathbf{w}})$ using U_{old} and X;
15 $\quad\quad Q$.addAll(W);
16 $\quad$ **end**
17 **end**
18 **return** X;

reuse, is thus highly effective when the variational single-objective solver can produce optimal solutions for the scalarized problem, as it can circumvent the *decoding* phase of variational algorithms, which is often very computationally intensive.

The single-objective variational solver (called on line 9) produces three outputs: the new reparameterized graph $\mathcal{U}'_{\mathbf{w}}$, an upper bound on the optimal scalarized payoff, $\bar{u}_{\mathbf{w}}$, and the approximally optimal joint action $\mathbf{a}_l$. Note that $\mathbf{a}_l$ implies a lower bound on the optimal payoff in $\mathbf{w}$, i.e., $\mathbf{w} \cdot \mathbf{u}(\mathbf{a}_l)$. All of these are stored (lines 10, 11 and 13).

If $\mathbf{u}(\mathbf{a}_l)$ is not already in X, then it is added to it and new corner weights are identified. Then, VOLS calculates the maximal possible improvement for those corner weights by solving a linear program (line 14) [7, 10]. Finally, the new corner weights are added to the priority queue Q (line 15). Because the maximal possible improvement to the scalarized payoff is guaranteed to be at one of the corner weights of X [2], VOLS terminates when Q is empty.

Upon termination, we can use U_{old} and X to determine the approximation quality ε, of X using the following corollary of Lemma 1:

Corollary 1. *VOLS returns X, an ε-CCS, where*

$$\varepsilon = \max_{(\mathbf{w}, \bar{u}_{\mathbf{w}}) \in U_{old}} \left(\bar{u}_{\mathbf{w}} - u^*_X(\mathbf{w}) \right).$$

4 Experiments

In this section, we compare the performance of VELS and VOLS on randomly generated MO-CoGs. For the single-objective subroutine, we use *weighted mini-buckets (WMB)* [4, 6], with an i-bound of $i = 1$. $i = 1$ is the highest degree of approximation. The MO-CoGs are generated following the procedure of Roijers et al. [10], which can produce a MO-CoG for any specified number of: n agents, d objectives, ρ local payoff functions, and $|\mathcal{A}_i|$ actions per agent. Starting from a fully connected graph with $n(n-1)/2$ local payoff functions, each of which connects two random agents, local payoff functions are removed randomly, until only ρ remain. An edge is removed only if doing so does not cut the graph. Finally, each local payoff function is filled with vectors of length d, containing real numbers drawn independently and uniformly from $[0, 10]$.

We compare VELS and VOLS on random 3-objective MO-CoGs with increasing numbers of agents n with $\rho = 1.8n$ factors per agent. We generated 25 MO-CoGs for each number of agents and ran both algorithms on the same instances. By increasing the number of agents and factors in this way, both the size of the joint action space and the *induced width* of the MO-CoG increase. Figure 2 (top left) shows that VELS is faster for the smallest possible problems. However, its runtime increases exponentially, whilst that of VOLS does not. For 55 or more agents, VOLS is faster than VELS. At 70 agents, VOLS is more than an order of magnitude faster, and at 150 agents VOLS is still faster than VELS is at 70 agents. To keep large MO-CoGs tractable, VOLS is thus highly preferable.

Of course, VOLS produces only an ε-CCS, whereas VELS produces an exact one. However, when we measure ε using Corollary 1, we find that it is consistently 1.1% of the value or smaller. In fact, in Figure 2 (top right), it even appears to decrease as a function of the size of the problem. At 150 agents, VOLS (with reuse) produced an ε-CCS with a ε of only 0.27% of the scalarized payoff. We thus conclude that VOLS' improved scalability comes at only a negligible cost in terms of payoff.

To test the effect of reuse on runtime, we compare the runtime of VOLS with and without reuse. We ran both versions on the same 25 instances for each number of agents. Figure 2 (top left) shows that VOLS with reuse requires consistenly less runtime across all numbers of agents. Across all numbers of agents, VOLS with reuse is a factor 1.22 faster. Figure 2 (bottom left), shows the ratio of the runtimes of VOLS with and without reuse (the runtime with reuse divided by the runtime without reuse), and the ratio of the ε produced by VOLS with and without reuse. While the runtime ratio gradually increases, meaning less benefit from reuse, the ε ratio gradually increases, meaning better accuracy. Furthermore, VOLS with reuse has lower runtime and ε overall. Hence, reuse contributes positively to VOLS' performance.

We also tested the effect of reuse on the runtime of the single-objective subroutine inside VOLS, using a single MO-CoG with $d = 3$, $n = 125$ and $\rho = 1.8n$. For each weight $\mathbf{w}$ in the sequence, we executed the variational subroutine with and without reuse. The average total runtime with reuse was $0.10s$ while it was $0.16s$ without reuse. Figure 2 (bottom right) shows the ratio between the runtime with and without reuse as a function of $\Delta\mathbf{w} = \mathbf{z} - \mathbf{w}$, i.e., the distance between the current weight on the weight on which the reused reparameterization is based. This figure shows that the runtime is positively correlated with $\Delta\mathbf{w}$. However, there are also a lot of weights for which reuse has little or effect, and even a few outliers for which reuse has a negative effect on the runtime. These outliers contribute disproportionaly to the average runtime: although they make up only 5% the weights, they are responsible for 48% of the total runtime of VOLS with reuse. For comparison, the first 5% of the calls, i.e., those with the 5% largest $\Delta\mathbf{w}$, account for only 9% of the runtime. An interesting direction for future work is to develop a method for identifying these outliers before executing the single-objective subroutine and then employing reuse only when it is expected to help.

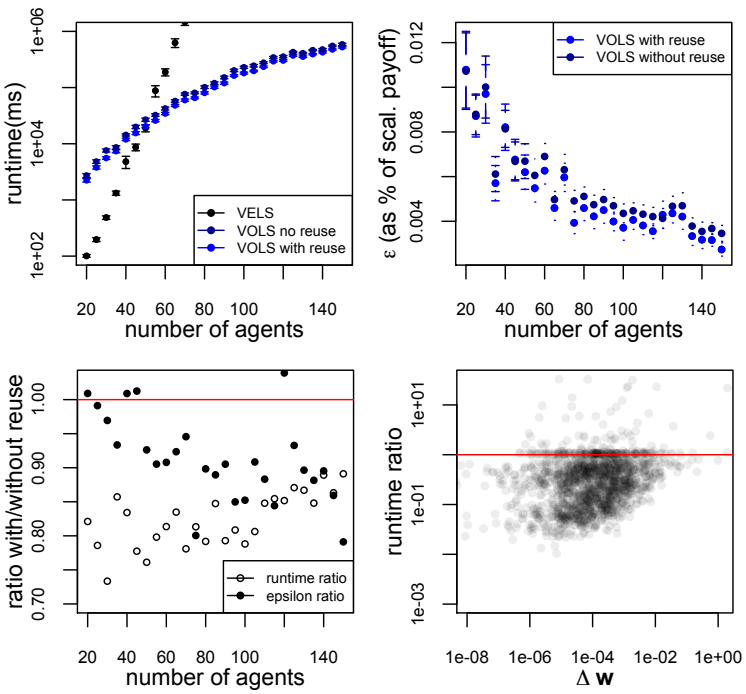

Figure 2: *(Top left)* The runtime (in logscale) of VOLS versus the runtime (in logscale) of VELS as a function of the number of agents n with $\rho = 1.8n$ and $d = 3$. The error bars represent SDOM. *(Top right)* The quality (ε) of the approximate CCSs produced by VOLS with and without reuse for the same MO-CoGs *(Bottom left)* The ratio of the runtimes and ε of VOLS with and without reuse for the same MO-CoGs *(Bottom right)* The runtime of the variational subroutine for different weights with reuse, divided by the runtime without reuse in logscale, as a function of the difference with the closest weight $\Delta\mathbf{w}$, for a MO-CoG with $n = 125$ with $\rho = 1.8$ and $d = 3$.

5 Conclusions

In this paper, we proposed *variational optimistic linear support (VOLS)*, a new method for finding a CCS for *multi-objective coordination graphs (MO-CoGs)*. To our knowledge this is the first vari-

ational method for MO-CoGs. VOLS solves a MO-CoG as a series of scalarized single-objective CoGs, for different scalarization weights $\mathbf{w}$. A key insight of VOLS is that the reparameterizations outputted by variational methods for earlier $\mathbf{w}$ in the series can be reused when the variational single-objective subroutine is called again for a similar new $\mathbf{w}$. Our experiments confirm that for large MO-CoGs, this reuse is key, and leads to both lower runtimes, and lower error. We therefore conclude that VOLS can efficiently solve large MO-CoGs that cannot be solved with exact methods, and that reparameterization reuse is a key component of the VOLS algorithm.

In future work, we aim to find a more efficient method, by analyzing and hopefully predicting the outliers of Figure 2 (bottom right). Furthermore, we aim to analyze the effect of the i-bound parameter of the WMB subroutine on the accuracy (in terms of ε) of VOLS.

Acknowledgments

This research is supported by NWO DTC-NCAP (#612.001.109) project, the NWO Innovational Research Incentives Scheme Veni (#639.021.336), and the NSF project #IIS-1254071. This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] H.-T. Cheng. *Algorithms for partially observable Markov decision processes*. PhD thesis, University of British Columbia, Vancouver, 1988.

[3] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 15 (NIPS'02)*, 2002.

[4] A. T. Ihler, N. Flerova, R. Dechter, and L. Otten. Join-graph based cost-shifting schemes. In *Proceedings of the Twenty-Eight Annual Conference on Uncertainty in Artificial Intelligence (UAI-12)*, pages 397–406, 2012.

[5] J. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, Dec. 2006.

[6] Q. Liu and A. T. Ihler. Bounding the partition function using Hölder's inequality. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 849–856, 2011.

[7] D. M. Roijers, J. Scharpff, M. T. J. Spaan, F. A. Oliehoek, M. M. de Weerdt, and S. Whiteson. Bounded approximations for linear multi-objective planning under uncertainty. In *ICAPS 2014: Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*, pages 262–270, June 2014.

[8] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 47:67–113, 2013.

[9] D. M. Roijers, S. Whiteson, and F. Oliehoek. Point-based planning for multi-objective POMDPs. In *IJCAI 2015: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1666–1672, July 2015.

[10] D. M. Roijers, S. Whiteson, and F. A. Oliehoek. Computing convex coverage sets for faster multi-objective coordination. *Journal of Artificial Intelligence Research*, 52:399–443, 2015.

[11] A. Rosenthal. Nonserial dynamic programming is optimal. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, pages 98–105. ACM, 1977.

[12] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1:219–254, 2011.

[13] P. Vamplew, R. Dazeley, E. Barker, and A. Kelarev. Constructing stochastic mixture policies for episodic multiobjective reinforcement learning tasks. In *Advances in Artificial Intelligence*, pages 340–349. 2009.

[14] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.