Queued Pareto Local Search for Multi-objective Decision Making[1]

Maarten de Waard[a] Maarten Inja[a] Chiel Kooijman[a]

Diederik M. Roijers[a] Shimon Whiteson[b]

[a] *University of Amsterdam, Informatics Institute*
[b] *University of Oxford, Department of Computer Science*

Abstract

Many real-world optimisation problems involve balancing multiple objectives. When there is no solution that is best with respect to all objectives it is often desirable to compute the *Pareto front*. Firstly, we propose *queued Pareto local search (QPLS)*, which improves on existing *Pareto local search (PLS)* methods by maintaining a queue of candidate solutions that prevents premature exclusion. Secondly, we propose *Pareto local policy search (PLoPS)*, which improves upon QPLS in the context of multi-objective sequential decision problems known as *multi-objective Markov decision processes (MOMDPs)*. PLoPS exploits the structure of MOMDPs by quickly identifying policies that are improvements without fully evaluating them, and intelligently initialising their values. We test the performance of QPLS and PLoPS on respectively *multi-objective coordination graphs (MO-CoGs)* and MOMDPs, and compare them to popular evolutionary and decision-theoretic alternatives. The results show that QPLS and PLoPS outperform these alternatives.

Real-world problems often have multiple possibly conflicting objectives. Solutions for such problems are evaluated for each objective and compared based on a *dominance* relationship such as the *Pareto dominance* relationship. In [1], we describe a new Pareto local search method called QPLS. QPLS is good at solving single-stage decision problems. However, some decision problems are sequential. We propose PLoPS [2] that improves upon QPLS in the context of sequential problems and exploits properties of our neighbourhood definition to use fewer and faster evaluations of MOMDP policies.

QPLS (Algorithm 1) finds a Pareto archive (set) P optimised w.r.t. a fitness function $\mathbf{f}$. Q is a queue initialized with random solutions. Until this queue is empty, these candidate solutions are popped one by one. When a solution π is obtained from the queue, the algorithm first runs a recursive *Pareto improvement* function (PI) at line 5. PI improves solutions by repeatedly selecting a dominating solution from the neighbourhood until no dominating neighbouring solution can be found. If the resulting solution π is undominated, k random incomparable solutions from π's neighbourhood are added to the queue. In existing algorithms solutions are often discarded because they are dominated by other solutions before their neighbourhood has been properly explored. We prevent this by employing a queue to only compare solutions after they have been improved fully by PI. QPLS, like other PLS algorithms, works well when tiny changes in an evaluated solution, i.e., the neighbourhood, can be evaluated quickly. To escape local minima QPLS can be embedded in a genetic scheme, yielding *Genetic QPLS* (GQPLS) [1].

Algorithm 1 QPLS$(Q, k, \mathbf{f})$

1: $P \leftarrow \emptyset$ ◀ *the Pareto archive*
2: **while** Q.notEmpty() **do**
3: $\quad \pi \leftarrow Q$.pop()
4: $\quad$ ▼ *recursive local improvements*
5: $\quad \pi \leftarrow$ PI$(\pi,\ \mathbf{f})$
6: $\quad$ ▼ π *undominated by P*
7: $\quad$ **if** $\forall p \in P : \mathbf{f}(\pi) \not\prec \mathbf{f}(p) \wedge \mathbf{f}(\pi) \neq \mathbf{f}(p)$
8: $\quad\quad P \leftarrow$ merge$(P,\ \{\pi\})$
9: $\quad\quad$ ▼ *new candidates*
10: $\quad\quad N \leftarrow \{\pi' \in \mathcal{N}(\pi) : \mathbf{f}(\pi) \not\prec \mathbf{f}(\pi')\}$
11: $\quad\quad Q$.addK$(N,\ k)$
12: $\quad$ **end if**
13: **end while**
14: **return** P

[1] Compressed contribution of the papers "Queued Pareto Local Search for multi-objective optimization" in *PPSN XIII*, pages 589-599. 2014. [1] and "Pareto local policy search for momdp planning" in *22th ESANN*, pages 53-58, 2015. [2]

In addition to balancing multiple objectives, real-world problems can also involve reasoning about future states and rewards. MOMDPs model such problems. An MOMDP is a tuple $\langle S, A, \mathcal{P}, \boldsymbol{\mathcal{R}}, \mu, \gamma \rangle$, where S is a finite set of states, A is a finite set of actions, $\mathcal{P} : S \times A \times S \rightarrow [0, 1]$ is a transition function that specifies, for each state and action, the probability of reaching the next state, and $\boldsymbol{\mathcal{R}} : S \times A \times S \rightarrow \mathbb{R}^D$ is the D-dimensional reward that specifies the immediate reward for each state, action and next state. μ denotes the probability distribution over the initial states and γ is a discount factor that specifies the relative importance of immediate rewards and future rewards. We consider the case where a solution to an MOMDP is a deterministic stationary policy π, that is, a mapping from states to actions.

PLoPS is a planning algorithm that extends QPLS to find a Pareto front of deterministic stationary policies for an MOMDP using new insights that allow it to require fewer evaluations and perform evaluations faster. Policies can be evaluated by *policy evaluation (PE)* as a fitness function $\mathbf{f}$. PE repeatedly applies Bellman backups to the value $\mathbf{V}^\pi(s)$ of each state s until convergence. This update is defined by:

$$\mathbf{V}^\pi_{t+1}(s) \leftarrow \sum_{s'} \mathcal{P}^{\pi(s)}_{ss'} \left[\boldsymbol{\mathcal{R}}^{\pi(s)}_{ss'} + \gamma \mathbf{V}^\pi_t(s') \right]. \tag{1}$$

PLoPS searches the *neighbourhood* $\mathcal{N}(\pi)$ of each policy π, which we define as the set of policies that can be made by changing one action for a single reachable state in π:

$$\mathcal{N}(\pi) = \left\{ \pi' \in \Pi : \exists s \in S^\pi \left[\pi(s) \neq \pi'(s) \ \wedge \forall (s' \neq s) \left[\pi(s') = \pi'(s') \right] \right] \right\}. \tag{2}$$

The PI function can be optimised for MOMDPs by quick evaluation of neighbouring policies. In [2] we show that a policy $\pi' \in \mathcal{N}(\pi)$ that is different from π only in performing a' rather than a in state s_c, Pareto dominates π iff:

$$\sum_{s'} \mathcal{P}^{\pi'(s_c)}_{s_c s'} \left[\boldsymbol{\mathcal{R}}^{\pi'(s_c)}_{s_c s'} + \gamma \mathbf{V}^\pi(s') \right] \succ \mathbf{V}^\pi(s_c). \tag{3}$$

Using (3), PLoPS can immediately identify policies as improvements by a single PE step on a single state, skipping all non-improving policies. Additionally, using the value of π as initialisation for the evaluation of π' allows PLoPS to find the value more quickly than standard heuristic initialisation. As similar policies often have similar values in large parts of the solution space, convergence is faster.

We empirically evaluated QPLS and PLoPS and compared them to popular alternatives. GQPLS was tested on a multi-objective coordination graph (MO-CoG), and compared to popular alternatives SPEA2 and NSGA-II, and GSPLS, a PLS method that does not use a queue. Figure 1 shows the Pareto fronts after one hour. GQPLS generates solutions that Pareto dominate those from other algorithms and have more diversity. PLoPS was tested on *Buridan's Ass (BA)*, a popular benchmark MOMDP which has a stochastic transition function and rewards in three dimensions. It was compared to MO-MCTS and NSGA-II in figure 2. As can be seen, PLoPS outperforms the other algorithms in time.

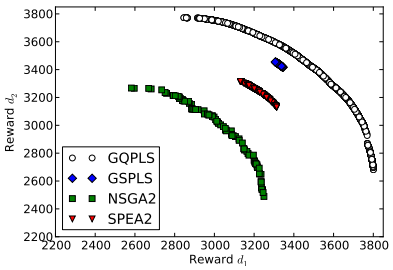

Figure 1: Pareto front for two-objective MO-CoG with 300 agents after one hour.

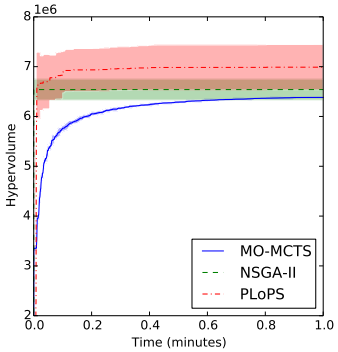

Figure 2: Mean and standard deviation of the hypervolume on Buridan's Ass with three objectives as a function of runtime.

In conclusion, QPLS employs a queue to quickly find a pareto set of optimal solutions (policies) outperforming popular alternatives. PLoPS improves upon QPLS in the context of MOMDPs and speeds up its policy evaluation by exploiting the structure of policies and their values. PLoPS speeds up its policy evaluation by exploiting the structure of policies and their values. Empirical results show that PLoPS performs well on benchmarks compared to popular alternatives.

References

[1] Maarten Inja, Chiel Kooijman, Maarten de Waard, Diederik M Roijers, and Shimon Whiteson. Queued pareto local search for multi-objective optimization. In *PPSN XIII*, pages 589–599. 2014.

[2] Chiel Kooijman, Maarten de Waard, Maarten Inja, Diederik M Roijers, and Shimon Whiteson. Pareto local policy search for MOMDP planning. In *22th ESANN*, 2015.