BLADE-BO: Cutting administrative burden using Quality Diversity Optimization Exam Scheduling at a Cornell University

Anonymous[1]

[1]Anonymous Institution

Abstract We present a novel framework Balanced Layered Active Diversity Exploration (**BLADE**) for final exam scheduling at Cornell University. We introduce an asynchronous, multi-fidelity extension of ParEGO (Knowles 2006) which leverages a Random Forest surrogate and a time-proportionate dispatch rule to balance low- and high-fidelity evaluations.

We created an interactive web-based front-end that enables stakeholders to filter and explore schedules through intuitive sliders and exclusion controls. Our approach integrates recent advances in Bayesian optimization (Hutter et al. 2011) with integer-programming-based timetabling (Ye et al. 2025).Using real co-enrollment data, we show that our algorithm efficiently explores the computationally expensive schedule-generation space and outperforms baselines—Thompson sampling (Thompson 1933) and Expected Improvement under Utility Uncertainty (EIUU) (Astudillo and Frazier 2020) on hypervolume benchmarks.

1 Introduction

Exam scheduling for large universities involves assigning hundreds of exams across limited time slots and venues while satisfying diverse preferences from students, faculty, and facilities (Carter et al. 1994). Traditional methods rely on optimization models such as integer programming (Ye et al. 2025), Constraint programming(Lee et al. 1996)., Simulated annealing algorithms (Bellio et al. 2021).

However, all these techniques require fixed weight assignments to different soft constraints, making it challenging to adjust and reoptimize schedules in response to stakeholder feedback without risking deterioration in other metrics. Repeated reweighting can be labor-intensive and may produce suboptimal trade-offs. In this paper, we propose **BLADE**-Balanced Layered Active Diversity Exploration. By balancing the time spent on each stage of our MIP we ensure a desirable balance between diversity and efficiency. We integrate this within an integer programming–based schedule generator adapted from (Ye et al. 2025) to rapidly produce candidate schedules under diverse constraints. Streamlining the decision-making process and reducing administrative burden.

2 Methods

We present an asynchronous, multi-fidelity, multi-objective Bayesian optimization algorithm employing a Random Forest surrogate to minimize a randomly-weighted objectives across sequential integer-program stages, balancing low- and high-fidelity evaluations using a time-proportionate dispatch rule.

We then upload plots, objectives, and parameters to a Faceted Search dashboard to allow University administrators to filter through schedules and express preferences.

2.1 Bayesian Optimization

This scheduling task is formulated as a multi-objective Quality-Diverse optimization problem with 14 decision variables $x \in \mathbb{R}^{14}$ and 8 objectives

$$f(x) = \big(f_1(x), \ldots, f_8(x)\big) \in \mathbb{R}^8.$$

As the algorithm runs, each schedule is randomly scalarized using a uniform random weight vector

$$w \sim \text{Dirichlet}(\underbrace{1, \ldots, 1}_{8 \text{ times}}), \quad w \in \Delta^8 = \left\{ w \in \mathbb{R}^8_{\geq 0} : \sum_{i=1}^{8} w_i = 1 \right\}.$$

The scalarized score for a sample x is

$$s(x; w) \;=\; \max_{i=1,\ldots,8} \; w_i \big(f_i(x) - \text{ref}_i \big),$$

where ref $\in \mathbb{R}^8$ is the fixed reference point which is *worse* than every schedule.

Surrogate fitting. After n evaluations (where $n \geq 2$) we have data $\mathcal{D}_n = \{(x_j, y_j)\}_{j=1}^{n}$ with $y_j = f(x_j)$. We normalize the x_j into $[0, 1]^{14}$ by $\tilde{x} = (x - \ell)/(u - \ell)$, then train a RandomForestRegressor $\hat{s}_n(\tilde{x})$ on the scalarized targets $s(x_j; w_j)$.

UCB-style Acquisition. To select the next evaluation point, we form a candidate pool $\mathcal{C}$ consisting of "low"-cost samples that reuse existing exam groupings and, if the remaining budget allows, an equal number of "high"-cost samples. We then employ an *active balancing* strategy: let

$$T_H = \sum_{\mathbf{x} \in \mathcal{S}_H} t(\mathbf{x}), \quad T_L = \sum_{\mathbf{x} \in \mathcal{S}_L} t(\mathbf{x})$$

denote the cumulative CPU time spent on our high- and low-cost evaluations, respectively. Given a target fraction $e \in [0, 1]$ for high-cost computation, we choose the next sample $\mathbf{x}^* \in \mathcal{C}$ to minimize

$$\left| \frac{T_H + t(\mathbf{x})}{T_H + T_L + t(\mathbf{x})} \;-\; e \right|,$$

where $t(\mathbf{x})$ is the estimated CPU time for $\mathbf{x}$. This criterion ensures that, after each iteration, the proportion of compute time devoted to high-cost samples remains as close as possible to the desired ratio e. Further motivation and empirical results appear in Section **??**.

We normalize them to $\tilde{X}_{\mathcal{C}}$, and compute per-tree predictions $\{p_{t,k}\}$. Let $\mu_k = \frac{1}{T} \sum_t p_{t,k}$, $\sigma_k = \sqrt{\frac{1}{T} \sum_t (p_{t,k} - \mu_k)^2}$. The acquisition value is

$$\alpha_k = \mu_k \;-\; \alpha \, \sigma_k,$$

We then pick the top option and dispatch it. The full pseudocode appears in Algorithm 1.

3 Experiments

We track 2 typical metrics in Multi-Objective Bayesian Optimization: Hypervolume and Sample Diversity.

Another metric specific to our setting is ensure a good balance between "High" and "low" cost evaluations. We want many Exam-Groupings to ensure there is a meaningful range of options for the registrar. We also observe that with more Exam-Groupings the model finds more diverse solutions. Many differences that are not captured by the 8 objectives we optimize can be important to the registrar (For example some exam groupings cause may cause room capacity problems).

It's important not to have too many "High" cost samples because they take at least 5 hours 12 minutes, so even with 5 samples being made in parallel that takes 104 hours for 100 samples, which is far too slow.

Our experiments show that CArBO's performance is acutely sensitive to both the budget allocation fraction—the share of total cost devoted to low-cost evaluations—and the number of fantasies generated by the acquisition function. In particular, as we increased the number of fantasies, we had to reduce the penalties for "High" cost evaluations, since drawing more low-cost fantasies makes it less likely that any high-cost candidate will achieve a two- or fivefold improvement. This difficulty motivated our choice to *Actively Balance* the compute time between stages.

3.1 Aquisition functions

Many Typical Bayesian Optimization techniques struggled to perform better than random sampling because the underlying exam-scheduling MIP landscape is highly non-smooth: tiny tweaks to parameters (e.g. block sizes or cost weights) can trigger completely different integer assignments and thus cause large, discontinuous jumps in all downstream metrics. Surrogate methods like ParEGO or EIUU presume a reasonably smooth, continuous response surface (or at least that random mixtures of objectives will average out noise), so their acquisition functions systematically misestimate which regions are actually promising.

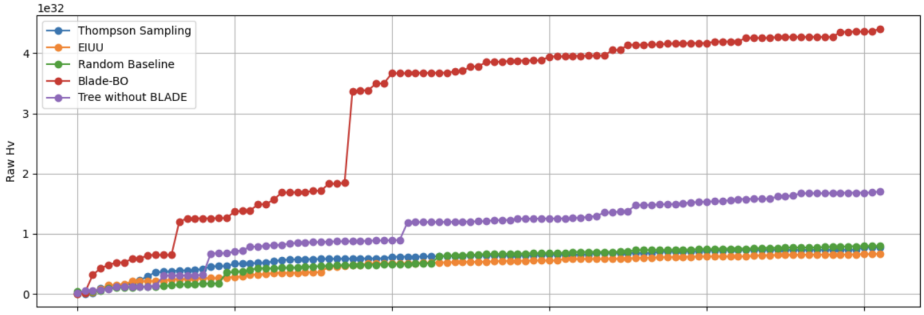


Figure 1: Acquisition functions comparison.

3.2 Importance of Exam-to-group Assignments

It is essential to produce enough high quality Exam-to-group Assignments to produce diverse high quality schedules.

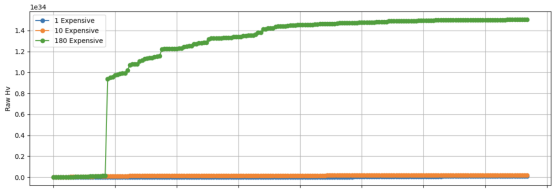


Figure 2: Hypervolume: 1,10, 189 expensive

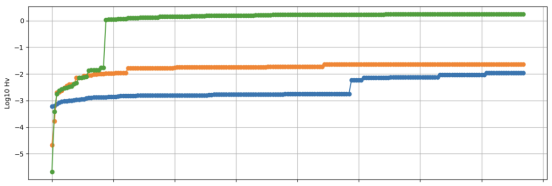


Figure 3: Log Hypervolume: 1,10, 189 expensive

This demonstrates the hypervolume progression is limited to - for a single Exam-to-group Assignment. It also shows that there is benefit in exploring many configurations for a single Exam-to-group Assignment. Therefore, finding a good balance between the "High" and "Low" cost samples is essential.

3.3 Model Debugging

To diagnose underperformance of certain acquisition functions, we trained both a SingleTaskGP and a RandomForest surrogate on all 2,724 experiments and evaluated their prediction accuracy and parameter sensitivity.

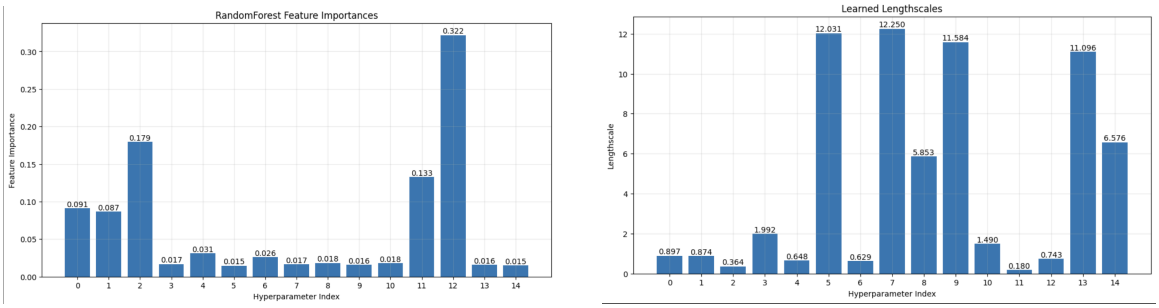
Prediction Accuracy

- **RandomForest** achieved a lower mean squared error (MSE) and mean absolute error (MAE = 194.64), indicating more reliable fitness estimates.
- **SingleTaskGP**, used by Thompson sampling and EIUU, exhibited substantially higher error (MAE = 194.64) and a corresponding MSE nearly double that of the tree model.

Parameter Sensitivity

- The Gaussian process's learned length scales disproportionately prioritized the Sequencing and Post-Processing parameters, despite their secondary role in determining overall schedule quality.
- In contrast, the RandomForest's feature importances closely mirrored the true parameter hierarchy, assigning the high weights to the first three Exam-Grouping parameters. It assigns the highest weights to the large_size_1 and large_cutoff_freedom parameters which we found were the most important in the past (Jovine et al. 2024)

Decision trees naturally handle non-smooth, hierarchical decision spaces and capture sequential dependencies more effectively (Hutter et al. 2011).



(a) Random Forest

(b) Single Task GP

Figure 4: Different Acquisition Functions learned feature importance

4 Future Work

1) 2 stage-> make it 3 stage 2) preference learning 3) Early stopping / adjusting time limits

5 Appendix

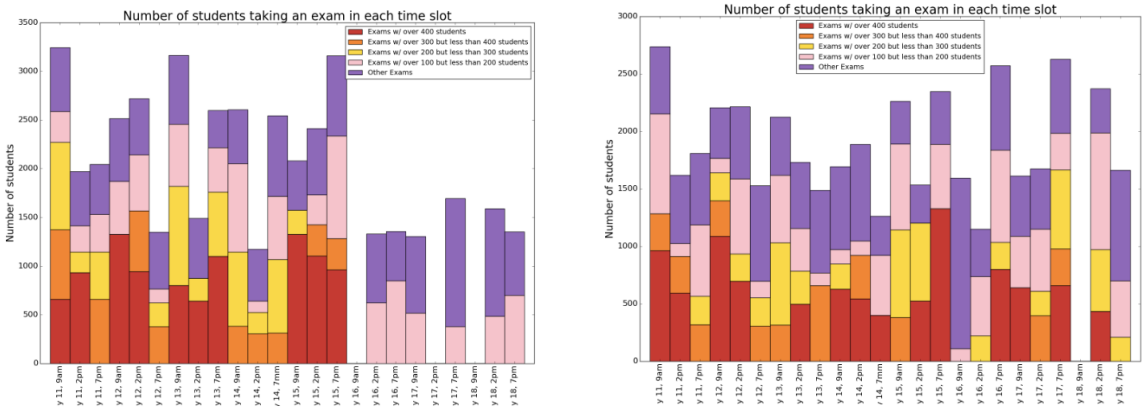
5.1 Scheduling Algorithm

We adapted the existing scheduling algorithm at Cornell (Ye et al. 2025). We must balance prioritizing the earlier scheduling of large-enrollment exams with minimizing back-to-back and triple-exam sequences. The exam scheduling algorithm proceeds in three stages.

1. **Exam-to-group assignment**: Exams are assigned to groups via a large IP that minimizes conflicts and lower-priority metrics (e.g., back-to-back exams). We solve this IP for 5 hours (though it can reach optimality in less time) with three parameters: `num_blocks` which is how many different times exam are scheduled, `reserved_blocks` which is how many exam times have no large exams, and `size_cutoff` how many students are taking an exam for it to be considered large (ranges from 200-300).

2. **Group sequencing**: Exam groups are sequenced to minimize back-to-back and triple exams. These parameters adjust objective function weights. α, γ, δ, θ, `large_block_size` is the minimum number of students for a block to be considered "large" (1000–2000). `large_exam_weight` penalizes scheduling large exams later. `large_block_weight` sets the overall importance of large-block considerations. `large_size_1` is the student-count threshold above which an exam is deemed large. `large_cutoff_freedom` controls how front-loaded large-exam blocks are: a value of 0 forces them to the front; 1 allows them to shift by one extra slot. These last two parameters have the greatest impact on the schedule's structure. (Jovine et al. 2024).

3. **Post Processing**: This takes subsets of exams and reschedules them to further improve metrics. This has 2 parameters `tradeoff` which is how much 2 exams in 24 hours is weighted relative to a back to back exam , `flpens` how much we weight having large exams in later slots.



(a) Schedule with cutoff freedom 0

(b) Schedule with cutoff freedom 4

Figure 5: Comparison of schedules under different cutoff freedom settings.

We track 8 objectives, all of which are minimized. In the exam-Grouping stage we also minmize conflicting exams. However, because this is such an important metric any groupings that have more than 5 conflicts are removed. Because all schedules have low conflicts it is not an objective . We have up to 3 exam slots per day, 7 to 9 final exam days. Metrics are evaluated sequentially and mutually exclusively, so a triple can't also be counted as two back-to-back exams.

Metric	Description
Triple Exams	Number of students with 3 exams in 24 hours
Three in four slots	Number of students with 3 exams in 4 slots
Back to backs	Number of students that have back-to-back exams
Two in threes	Number of students with 2 exams in 3 slots
Singular late exams	Number of students with 1 exam on the last 3 days
Two exams, large gap	Number of students with a gap of more than 5 days between exams
Average Max	Average day of the student's last exam
Lateness	Weighted sum of the number of students in large exams in late slots

5.2 Frontend

We implemented a Frontend using React. it allowed users to perform a Faceted Search over the schedules produced. The frontend allows filtering on parameter, metrics, and exam days. It also allows the user to pin and download schedules.

5.3 Experimental Setup

The experiments were configured as follows:

- **Hardware and runtime**: 100 iterations, maximum runtime 10 hours, on 5 CPUs (`Intel(R) Xeon(R) Silver 4214R @ 2.40 GHz`), 1 GPU (`NVIDIA GeForce RTX 3090`), and 128 GB RAM, on Cornell's G2 cluster.

Figure 6: Frontend

- **Reference point:**

$$\text{REF\_POINT} = \begin{pmatrix} \text{Triples} : 250, \\ \text{3 in 4s} : 600, \\ \text{Back-to-backs} : 3000, \\ \text{Triples} : 5000, \\ \text{Singular late exams} : 1500, \\ \text{2 exams big gap} : 1200, \\ \text{Average maximum} : 20, \\ \text{Lateness} : 60000 \end{pmatrix}$$

- **Stage time limits:**

$$\text{Exam} - \text{Grouping} - \text{Time} = 5{\times}3600, \quad \text{Sequeincing} - \text{TIME} = 360, \quad \text{Post} - \text{Processing} - \text{TIME} = 360.$$

- **Parameter bounds:**

$$\begin{aligned} \texttt{BLOCK\_BOUNDS} &= \{\, \text{size\_cutoff} : [200, 300],\ \text{reserved} : [0, 7],\ \text{num\_blocks} : [20, 24] \,\}, \\ \texttt{SEQ\_BOUNDS} &= \{\, \alpha : [10, 100],\ \gamma : [5, 50],\ \delta : [0, 10],\ : [0, 10],\ \theta : [0, 10], \\ &\qquad \text{large\_block\_size} : [1000, 2000],\ \text{large\_exam\_weight} : [0, 100], \\ &\qquad \text{large\_block\_weight} : [0, 100],\ \text{large\_size\_1} : [200, 400],\ \text{large\_cutoff\_freedom} : [0, 4] \,\}, \\ \texttt{PP\_BOUNDS} &= \{\, \text{tradeoff} : [0, 1],\ \text{flpens} : [0, 1] \,\}. \end{aligned}$$

5.4 Aquisition Function Experiments

Warm-start initialization: Five high-cost samples were generated in advance to initialize all runs, improving fairness and reducing startup overhead.

5.5 Pseudo-Code

Algorithm 1 Cost-Aware Multi-Fidelity Tree-Based Optimization

Require: bounds ℓ, u, iterations N, pool size q, UCB weight α, expensive ratio r

$X \leftarrow [\,], Y \leftarrow [\,], t_{\text{low}} \leftarrow 0, t_{\text{high}} \leftarrow 0$
start thread pool with q workers ▷ Initial low-fidelity samples
for $i = 1$ to 5 **do**
    $x \leftarrow$ `sample_existing(1)`
    `dispatch_evaluation(`x, is_high = False`)`
end for ▷ Main optimization loop
$n \leftarrow 0$
while $n < N$ **do**
    $(x, y, c, h) \leftarrow$ `wait_for_completion()`
    append x to X, append y to Y
    if h **then** t_{high} += c
    elset_{low} += c
    end if
    n += 1 ▷ Fit surrogate model
    draw random $w \in \mathbb{R}^m_{\geq 0}$, normalize so $\sum_i w_i = 1$
    $s_j \leftarrow \max_i[w_i \cdot (Y_{j,i} - \text{ref}_i)]$ for each row Y_j
    $X_{\text{norm}} \leftarrow (X - \ell)/(u - \ell)$
    train RF $\leftarrow$ `RandomForestRegressor()` on (X_{norm}, s) ▷ Generate candidates and compute acquisition
    $C_{\text{low}} \leftarrow$ `sample_existing(`k`)`
    $C_{\text{high}} \leftarrow$ `sample_new(`k`)` if time budget remaining, else $[\,]$
    $\mu_t \leftarrow$ `tree.predict(`C`)` for each tree t in RF
    $\mu \leftarrow \text{mean}(\mu_t), \sigma \leftarrow \text{std}(\mu_t)$
    $\text{acq} \leftarrow \mu - \alpha \cdot \sigma$ ▷ Choose fidelity based on time balance
    $f \leftarrow t_{\text{high}}/(t_{\text{low}} + t_{\text{high}})$
    if $f < r$ **and** $C_{\text{high}} \neq [\,]$ **then**
        $x^* \leftarrow \arg\max \text{acq}[C_{\text{high}}]$
        `dispatch_evaluation(`x^*, is_high = True`)`
    else
        $x^* \leftarrow \arg\max \text{acq}[C_{\text{low}}]$
        `dispatch_evaluation(`x^*, is_high = False`)`
    end if
end while
$M \leftarrow$ `pareto_mask(`Y`)`
return $X[M], Y[M]$

References

Raul Astudillo and Peter I. Frazier. Multi-attribute bayesian optimization with interactive preference learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108, pages 4496–4507. PMLR, 2020.

Ruggero Bellio, Sara Ceschia, Luca Di Gaspero, and Andrea Schaerf. Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling. *Computers & Operations Research*, 132:105300, 2021.

Michael W. Carter, Gilbert Laporte, and John W. Chinneck. A general examination scheduling system. *Interfaces*, 24(3):109–120, 1994.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization (LION-5)*, pages 507–523. Springer, 2011. doi: 10.1007/978-3-642-25566-3_40.

Adam Jovine, Jian Kai Ang, Ariel Stern, Esha Shah, Thomas Gambra, and David Shmoys Cecilia Yang. Investigating the tradeoffs of different final exam structures. Poster presentation at the Cornell Undergraduate Research Symposium, Cornell University, Ithaca, NY, April 2024. URL `https://docs.google.com/presentation/d/1b25y4wrr1_LTyM6eJ_4klgOfcH10qQGytJRu-u3mZxg/edit?slide=id.g2cbefac699a_1_5#slide=id.g2cbefac699a_1_5`.

Joshua D. Knowles. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006. doi: 10.1109/TEVC.2005.852430.

Eric Hans Lee, Valerio Perrone, Cédric Archambeau, and Matthias Seeger. Constraint logic programming for examination timetabling. *The Journal of Logic Programming*, 26(2):217–233, 1996.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

Tinghan Ye, Adam Jovine, Willem van Osselaer, Qihan Zhu, and David B. Shmoys. Universidad politécnica de madrid uses integer programming for scheduling weekly assessment activities. *INFORMS Journal on Applied Analytics*, 56(2):1–40, 2025.