Optimization approaches

Short answer: **Yes — once you have a concrete success metric, formulating placement as an optimization (typically integer/linear) is a very sensible next step.**  
Below I’ll give a practical prescription: what the optimization looks like, how to get the inputs (you need predicted/causal gains per student-course), constraints to include, solver/implementation choices, how to deal with uncertainty and fairness, and a short toy mathematical formulation you can adapt.

**1) Why optimization is the right next step**

Prediction/causal models tell you *how likely a given student is to succeed* in each course (or how much benefit they'd get from being placed into a particular course). But your **operational objective** — assigning real students to a limited number of seats — is an allocation problem. Optimization lets you translate predicted benefits and constraints (seat counts, prerequisites, fairness, instructor loads) into a single, globally optimal assignment.

**2) What you MUST define first**

You said you don’t yet have success metrics — you need at least one concrete objective to optimize. Examples:

* Maximize total probability of *passing* first-term math (sum of probabilities).
* Maximize expected total grade-points (sum of expected grades).
* Minimize the number of “mismatches” (students placed above/below their predicted successful level).
* Maximize equity-adjusted success (weighted success that boosts disadvantaged groups).
* Minimize expected remediation downstream or cost per success (cost-aware objective).

Pick one (or a weighted multi-objective) — everything below depends on that choice.

**3) Inputs to the optimization (where ML/causal fits)**

You need a numeric **value v\_{i,j}** for each student i and course j representing the objective contribution if i is assigned to j.

Two ways to get v\_{i,j}:

1. **Predictive approach** — train a predictive model P(Y=success | X, course=j). Use XGBoost or logistic regression to estimate the probability of success for student i in course j. Then set v\_{i,j} = P̂(success | i,j) (or expected grade).
   * Caveat: these are associative estimates — good for predicting under *current* policies but not necessarily causal if assignment historically depended on unobserved factors.
2. **Causal approach (preferred for interventions)** — estimate the **causal effect** or expected outcome under assignment using RCT / causal forest / doubly robust methods.
   * For placements, the quantity you want is typically the expected outcome **if** student i were placed in course j (potential outcome). Use causal estimators / meta-learners or RCT data to estimate μ̂\_j(X\_i) or τ̂ between courses.
   * Set v\_{i,j} = μ̂\_j(X\_i) (expected success under j). If you have a baseline course b, you can let v\_{i,j} = μ̂\_j(X\_i) − μ̂\_b(X\_i) (uplift vs baseline) and include baseline assignment constraints.

Bottom line: **use causal estimates for v\_{i,j} whenever possible** — otherwise your allocation may propagate past assignment biases.

**4) Example integer linear program (ILP) formulation**

Notation:

* i = students (i = 1..N)
* j = course sections (j = 1..M)
* x\_{i,j} ∈ {0,1} indicator: 1 if student i assigned to course j, else 0
* v\_{i,j} = value (expected success/probability/grade) for student i in course j
* cap\_j = capacity (number of seats) in course j

Basic assignment objective (maximize total expected success):  
maximize Σ\_{i=1..N} Σ\_{j=1..M} v\_{i,j} x\_{i,j}

subject to:

* Each student assigned to exactly one placement (or to at most one, depending):  
  Σ\_{j} x\_{i,j} = 1, ∀ i
* Capacity limits:  
  Σ\_{i} x\_{i,j} ≤ cap\_j, ∀ j
* Eligibility / prerequisites:  
  x\_{i,j} = 0 for ineligible pairs (or add constraints to force 0)
* Optionally, integer constraint: x\_{i,j} ∈ {0,1}

You can add more constraints (below).

**5) Typical practical constraints & extensions**

* **Allow multiple placements per student** (e.g., course + co-req): model with separate decision variables or multi-dimensional assignment.
* **Capacity & instructor load**: capacity is per section; you may also limit total load per instructor.
* **Minimum or maximum allocations by group (fairness):** e.g., ensure at least α fraction of seats for first-gen students in support course, or limit disparity in expected success between groups. These are linear constraints.
* **Cost constraints:** If some placements are more expensive (TA support, smaller section), include cost c\_{j} and a budget constraint Σ\_{i,j} c\_j x\_{i,j} ≤ Budget. Or optimize net benefit Σ v\_{i,j} x\_{i,j} − λ Σ c\_j x\_{i,j}.
* **Soft constraints / penalty terms:** If you prefer not to hard-constrain fairness but to penalize violations, incorporate penalties in objective (still linear if penalties are linear).
* **Prerequisites / sequencing:** If assignment affects future semesters, this becomes multi-period optimization (stochastic dynamic programming). Start with single-term static assignment, then extend if needed.
* **Interference / class composition:** success can depend on who else is in class. Then v\_{i,j} is not separable — model becomes combinatorial (nonlinear). Approaches: approximate by using expected peer effects or include class-level features in v\_{i,j} estimated under likely compositions; or use iterative assignment with re-estimation.

**6) Solvers & implementation suggestions**

* **R:** ompr + ompr.roi (with solver ROI/CBC), lpsolve, Rsymphony (wrapper for SYMPHONY), or call gurobi / cplex if available.
* **Python:** PuLP, OR-Tools (Google), Pyomo (with CBC/GLPK/Gurobi).
* **For assignment/min-cost flow problems:** treat as min-cost max-flow (faster) — NetworkSimplex algorithms available in OR-Tools.
* **Scale:** ILPs with thousands of students × few course options are usually solvable quickly with CBC or OR-Tools. If M×N gets huge or objective becomes nonlinear, consider heuristics (greedy, local search) or relax to LP and then round.
* **Open-source recommendation:** start with ompr (R) or PuLP (Python) + CBC.

**7) Uncertainty & robustness**

* Your v\_{i,j} estimates have error. Options:
  + **Robust optimization:** optimize for worst-case within an uncertainty set for v\_{i,j} (conservative).
  + **Stochastic programming:** include scenarios for v (sample from posterior predictive distribution) and optimize expected value across scenarios (two-stage stochastic).
  + **Regularize assignment:** limit how different the new assignment can be from current assignment to reduce risk.
  + **Bootstrap evaluation:** simulate assignment using your model, estimate realized outcomes under uncertainty to measure expected gains and risk.
* Practical approach: produce the deterministic ILP assignment using point estimates, then perform offline policy evaluation (DR/IPS) and uncertainty quantification to see how sensitive the solution is. If fragile, prefer pilot/RCT.

**8) Operational & ethical safeguards**

* **Allow advisor overrides** (track them — they reveal model shortcomings).
* **Cap changes per term** (don’t reassign huge fractions at once).
* **Fairness constraints** (statistical parity, equalized opportunity) — include as linear constraints or penalties.
* **Explainability:** provide simple explanations or decision rules for students/advisors (policy trees or small interpretable policy approximations).
* **Pilot & monitoring:** run a randomized pilot or A/B test the policy before full rollout.

**9) Simple worked example (mathematical + narrative)**

Suppose:

* Objective: maximize total probability of passing first-term math.
* For each student i and course j you have P̂\_{i,j} = estimated probability of pass (from causal model).
* Each student must have exactly one placement and each section j has cap\_j seats.

ILP:  
maximize Σ\_i Σ\_j P̂\_{i,j} x\_{i,j}  
s.t.  
Σ\_j x\_{i,j} = 1 ∀ i  
Σ\_i x\_{i,j} ≤ cap\_j ∀ j  
x\_{i,j} ∈ {0,1}

Solve with CBC (ompr or PuLP) and inspect assignment. Then compute expected gain = Σ\_i P̂\_{i,assigned(i)} − Σ\_i P̂\_{i,current\_assignment(i)}. Use A/B pilot to validate.

**10) Next practical steps I’d recommend you take now**

1. **Define one or two success metrics** you care about (pass probability, expected grade, or downstream retention).
2. **Estimate v\_{i,j}** using best causal method you can (RCT if available; otherwise DR/causal forests/meta-learners using XGBoost for nuisance estimates).
3. **Formulate the ILP** as above, adding fairness/budget/prereq constraints that matter to you.
4. **Solve offline** and evaluate policy value with DR/IPS to estimate counterfactual gains and uncertainty.
5. **Pilot** with randomized rollout (stratified) and monitor.

If you want, I can **do one of the following next** (pick one):

* Draft a concrete ILP for your real variables (I’ll assume an objective you pick — tell me which metric), including fairness and budget constraints.
* Provide a small R script using ompr that builds and solves the toy ILP from the example.
* Show how to convert XGBoost/causal-forest outputs into the v\_{i,j} matrix and then plug that into an optimizer.

Which would you like me to produce right now?