Storage Location Assignment with Mergeable Locations

Ermal Belul[1,2], Marwane Bouznif[2], Dritan Nace[1], Antoine Jouglet[1]

[1] Université de Technologie de Compiègne, Rue Roger Couttolenc, 60200 Compiègne, France
{ermal.belul@hds.utc.fr, dritan.nace@hds.utc.fr, antoine.jouglet@hds.utc.fr}
[2] Savoye
{ermal.belul@savoye.com, marwane.bouznif@savoye.com}

Keywords : *Storage Location Assignment Problem, Max-Flow Min-Cost, Logistics*

1 Introduction

The Storage Location Assignment Problem (SLAP) is a key decision in warehouse design, as it drives both picking effort and replenishment frequency [3]. In [1], we modeled SLAP as a cost-weighted assignment between products and locations and solved it as a min-cost flow problem on large industrial instances. In many automated systems, neighboring locations (pallet, dynamic or locker positions) can be merged to store more units of the same product. Merging preserves geometric feasibility but increases available volume, thereby reducing replenishments while consuming more slots and possibly moving stock to less favorable picking positions.

We propose a merge-aware SLAP in which *mergeable neighbor groups* are explicitly modeled. Our flow-based heuristic combines the baseline method of [1] with a greedy activation of mergeable groups and a cost-bounded MILP polishing phase, in order to increase effective storage usage while making sure to not surpass the total cost calculated on the classical SLAP.

2 Presented approach

We consider products P and atomic locations L. Each product $i \in P$ has average picking frequency f_i and demand volume; each location $j \in L$ has capacity V_j and picking cost coefficient c_j^{pick}. For each feasible pair (i, j) we define $C_{ij} = \alpha\, c_j^{\text{pick}} f_i + \beta\, c_{ij}^{\text{repl}}$, where c_{ij}^{repl} estimates the replenishment effort when product i is stored in location j, and α, β weight the two criteria.

From layout data we infer a set G of mergeable groups $g \subseteq L$, each composed of mutually neighboring locations. A group g has total volume $V_g = \sum_{j \in g} V_j$ and uses the cheapest location in g as access point $j^\star(g)$. The assignment cost of product i to group g is computed as $C_{ig}^{\text{group}} = \alpha\, c_{j^\star(g)}^{\text{pick}} f_i + \beta\, c_{ig}^{\text{repl}}(V_g)$, with the same feasibility rules as for atomic locations.

Baseline min-cost flow. The baseline approach is the same as in [1], so we ignore merging and build bipartite network between products and locations, with capacity-one arcs (i, j) and cost C_{ij} . To solve the min cost max flow problem in a bipartite graph we use the Hungarian algorithm [2], which runs in $O(n^3)$ time. This yields a reference flow $F^\star$ (all assignable products are placed) and a minimum cost $C^\star$, which serve as feasibility and cost benchmarks for our merge-aware extension.

Greedy activation of mergeable groups. We extend the baseline network by adding a node for each group $g \in G$ and arcs: (i) (s, g) from the source with capacity $|g| - 1$, (ii) (g, j) to its locations $j \in g$, and (iii) (i, g) from eligible products with cost C_{ig}^{group}. Groups are sorted by a simple score based on their cheapest picking position and activated one by one. After each activation we recompute the maximum flow; if it remains equal to $F^\star$ (no product becomes unassigned) the group is kept, otherwise it is discarded. This yields a feasible extended network with an associated flow, which we use as a warm start for a MILP.

MILP polishing under a cost constraint. The final step is a mixed-integer linear model defined on the extended network. Each arc a has an integer flow variable x_a (bounded by its capacity) and the objective is to maximize the total flow leaving the source, interpreted as the effective number of exploited locations. Standard flow conservation is enforced at every intermediate node, and additional coupling constraints ensure that activating a group of size $|g|$ consumes $(|g| - 1)$ extra slots and is consistent with the product flows entering the group. The model includes a global cost constraint $\sum_a c_a x_a \leq (1 + \varepsilon) C^\star$, where ε is a small tolerance (1.5% in our experiments). The min-cost flow from the greedy-activated network initializes the variables x_a, enabling the MILP solver to focus on improving flow while respecting the cost bound.

3 Results and discussion

Figure 1 synthesizes the behaviour of the different solution strategies. The no-merge min-cost flow provides the classical SLAP benchmark. The greedy-merge min-cost flow slightly increases the total flow but also yields a higher total cost than the baseline. This is expected: by merging some of the best locations (low c_j^{pick}) to host high-turnover items, the algorithm reduces replenishment requirements but simultaneously depletes the pool of low-cost positions, forcing other products into more expensive locations. The net effect at this stage is therefore an increase in the weighted picking–replenishment cost, but this step is needed only to provide a feasible MIP start for the polishing phase.

The MILP polishing phase, run under different time limits, progressively activates more mergeable groups and produces solutions that dominate both the baseline and the greedy assignment in the cost–flow plane: total cost decreases while flow increases. Beyond this region, as the time limit grows and more groups are activated, the marginal savings in replenishment are outweighed by the loss of cheap picking locations, and the total cost starts to rise again even though the flow continues to increase. In our experiments, the best trade-off between cost reduction and increased flow is obtained around the intermediate time limit (60 seconds), which can be interpreted as a “sweet spot” on the empirical cost–flow frontier generated by the merge-aware SLAP.

Label	Time (s)	Flow	Cost	Merges
Baseline SLAP	9.0	1920	645849	0
Greedy Merge	276	1925	649005	5
Hexaly (10)	10.0	1930	640688	5
Hexaly (30)	30.0	1937	639368	12
Hexaly (60)	60.0	1955	638491	30
Hexaly (120)	120.0	1979	639786	54
Hexaly (360)	360.0	1994	642318	69

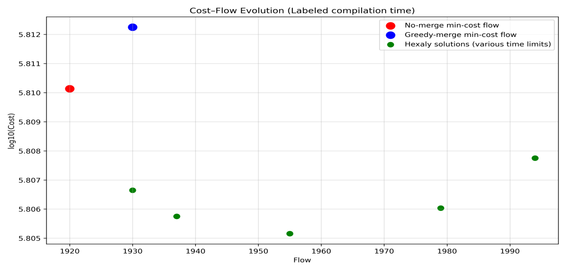

FIG. 1: Left: Numerical Results. Right: cost–flow evolution ($\log_{10}$ scale for cost).

References

[1] E. Belul, D. Nace, A. Jouglet, and M. Bouznif. Strategic solutions for warehouse optimization: Tackling the Storage Location Assignment Problem. In *25e congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2024)*, Amiens, France, 2024.

[2] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52(1):7–21, 2005.

[3] G. Richards. *Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse.* Kogan Page Publishers, 2017.