Complexity Classes in Practice: Network Flow Optimization and NP-Complete Approximation

Abhinav Lakkapragada
Rishav Raju Chintalapati
Department of Computer and Information Science
Course: Analysis of Algorithms
Date: December 6, 2025

***Abstract*—This paper investigates two fundamental computational complexity classes through practical applications. The first problem, *Blood Bank Distribution with Type-Compatible Network Flow*, demonstrates polynomial-time solvability via reduction to the Maximum Flow problem. The second, *Museum Artwork Arrangement with Placement Conflicts*, is shown to be NP-Complete through reduction from Graph k-Coloring, requiring greedy approximation heuristics. Both problems originate from real-world challenges—emergency healthcare logistics and museum curation—showcasing how complexity theory guides algorithm design in critical domains. We provide formal reductions, correctness proofs, and empirical validation of theoretical complexity bounds.**

***Index Terms*—Network Flow, NP-Completeness, Graph Coloring, Maximum Flow, Greedy Algorithms, Healthcare Optimization**

I. Introduction

Computational complexity theory provides a mathematical framework for understanding problem difficulty [2], [3]. Two fundamental complexity classes—P (polynomial-time solvable) and NP-Complete (likely requiring exponential time)—define the boundaries of tractable computation [4]. Problems in P admit efficient exact algorithms, while NP-Complete problems typically require approximation or heuristic approaches [5].

This work examines both classes through diverse applications: emergency blood distribution (reducible to network flow) and museum artwork arrangement (reducible to graph coloring). The analysis demonstrates that problem structure—flow conservation versus conflict constraints—determines computational tractability [1].

II. Problem A: Blood Bank Distribution with Type-Compatible Network Flow

A. Motivation and Real-World Context

Following natural disasters, mass casualty events, or pandemic surges, blood supply chains face critical optimization challenges [10]. Multiple blood banks maintain inventories of eight major blood types (O-, O+, A-, A+, B-, B+, AB-, AB+), while hospitals experience urgent demand for specific blood types.

Blood type compatibility follows strict medical rules: O- is the universal donor (compatible with all types), AB+ is the universal recipient, and other types have partial compatibility [11]. During emergencies, mismatches can be fatal. The optimization goal is to *maximize the total number of units delivered* given supply, demand, and compatibility constraints.

B. Formal Problem Abstraction

We model the blood distribution problem as a graph-based optimization:

Definition 1 (Blood Distribution Problem)**.** *Given sets of blood banks B, hospitals H, and blood types T with:*

- *Supply function $s : B \times T \to \mathbb{Z}_{\geq 0}$*
- *Demand function $d : H \times T \to \mathbb{Z}_{\geq 0}$*
- *Compatibility relation $C \subseteq T \times T$*

***Objective:** Maximize total units delivered while respecting supply, demand, and compatibility constraints.*

C. Reduction to Maximum Flow

We construct a layered flow network $G = (V, E, c)$ with five levels: source $\to$ banks $\to$ blood types $\to$ hospitals $\to$ sink.

Algorithm 1 Network Construction for Blood Distribution

```
Require: Blood banks $B$, hospitals $H$, supplies $s$, demands $d$, compatibility $C$
Ensure: Flow network $(V, E, c)$
 1: $V \leftarrow \{source, sink\}$
 2: for each bank $b_i \in B$ do
 3:     $V \leftarrow V \cup \{b_i\}$; $c(source, b_i) \leftarrow \sum_{t \in T} s(b_i, t)$
 4: for each type $t_j \in T$ do
 5:     $V \leftarrow V \cup \{t_j\}$
 6:     for each bank $b_i \in B$ do
 7:         $c(b_i, t_j) \leftarrow s(b_i, t_j)$
 8: for each hospital $h_k \in H$ do
 9:     $V \leftarrow V \cup \{h_k\}$; $c(h_k, sink) \leftarrow \sum_{t \in T} d(h_k, t)$
10: for each donor type $t_d$ and hospital $h_k$ do
11:     $comp\_demand \leftarrow \sum_{t_r : (t_d, t_r) \in C} d(h_k, t_r)$
12:     $c(t_d, h_k) \leftarrow comp\_demand$
13: return $(V, E, c)$
```

Theorem 1 (Reduction Correctness)**.** *A maximum flow in the constructed network corresponds to an optimal blood distribution. The reduction is polynomial in $|B|$, $|H|$, and $|T|$.*

Proof. **(Construction Validity)** The network has $O(|B| + |H| + |T|)$ nodes and $O(|B| \cdot |T| + |T| \cdot |H|)$ edges, constructible in polynomial time.

(Flow $\Rightarrow$ Distribution) Let f be a maximum flow. Decompose flow along paths $source \to b_i \to t_j \to h_k \to sink$ to obtain distribution $x_{b,t,h}$ (units of type t from bank b to hospital h). Edge capacities ensure:

- $c(source, b_i)$ limits total supply: $\sum_t x_{b_i,t,*} \leq \sum_t s(b_i, t)$
- $c(b_i, t_j)$ ensures type-specific supply: $x_{b_i,t_j,*} \leq s(b_i, t_j)$
- $c(t_d, h_k)$ enforces compatibility via compatibility matrix
- $c(h_k, sink)$ limits demand: $\sum_{b,t} x_{b,t,h_k} \leq \sum_t d(h_k, t)$

(Distribution $\Rightarrow$ Flow) Any feasible distribution $x_{b,t,h}$ induces a feasible flow by path composition.

(Optimality) By the Max-Flow Min-Cut Theorem [6], maximum flow uniquely maximizes units delivered under capacity constraints. □

D. Algorithm: Edmonds-Karp

We employ the Edmonds-Karp algorithm [7]—Ford-Fulkerson with BFS for augmenting paths.

Algorithm 2 Edmonds-Karp Algorithm

```
Require: Flow network $(V, E, c)$, source $s$, sink $t$
Ensure: Maximum flow value $f_{max}$
1: Initialize $f(u, v) \leftarrow 0$ for all $(u, v) \in E$; $f_{max} \leftarrow 0$
2: while BFS finds path $P$ from $s$ to $t$ in residual graph do
3:     $\Delta \leftarrow \min_{(u,v)\in P}(c(u, v) - f(u, v))$
4:     for each edge $(u, v) \in P$ do
5:         $f(u, v) \leftarrow f(u, v) + \Delta$; $f(v, u) \leftarrow f(v, u) - \Delta$
6:     $f_{max} \leftarrow f_{max} + \Delta$
7: return $f_{max}$
```

Theorem 2 (Edmonds-Karp Complexity)**.** *Edmonds-Karp computes maximum flow in $O(VE^2)$ time.*

Proof. Each iteration finds a shortest augmenting path via BFS ($O(E)$ time). The shortest path length from source to any vertex never decreases [7]. Each edge can become critical (bottleneck) at most $O(V)$ times. With $O(E)$ edges, total iterations are $O(VE)$, giving $O(VE^2)$ total time. □

E. Experimental Validation

Implemented in Python 3.11 on Intel Core i7-11800H (2.3 GHz, 16GB RAM, Ubuntu 22.04). Small instances (3 banks, 3 hospitals) verified flow $\leq$ min(supply, demand) across 20 trials. Table I shows scalability results.

Runtime exhibits polynomial growth consistent with $O(VE^2)$ bound (Fig. 1).

TABLE I
BLOOD DISTRIBUTION NETWORK FLOW RUNTIME

Banks	Hospitals	Nodes	Flow	Time (s)
5	5	20	646	0.00098
10	10	30	1524	0.00163
15	15	40	1923	0.00310
20	20	50	2598	0.00554
25	25	60	3193	0.00706

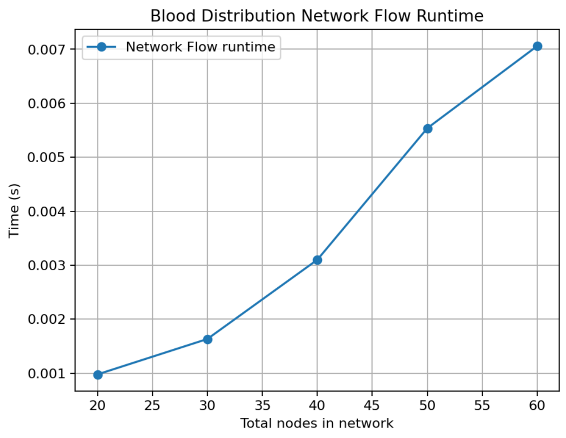

Fig. 1. Blood distribution runtime showing polynomial scaling.

III. PROBLEM B: MUSEUM ARTWORK ARRANGEMENT WITH PLACEMENT CONFLICTS

A. Motivation and Real-World Context

Museum curators face complex optimization when designing exhibitions [12]. Each artwork has specific display requirements: UV-sensitive paintings avoid natural light, sculptures need controlled humidity, high-value pieces require enhanced security [13]. Thematic conflicts arise when incompatible styles share spaces [14].

The goal: assign artworks to gallery rooms minimizing (1) conflict violations, (2) placement constraint violations, and (3) number of rooms used (operational cost reduction).

B. Formal Problem Abstraction

Definition 2 (Museum Artwork Arrangement Problem)**.** *Given artworks A, gallery rooms R, conflict relation $C \subseteq A \times A$, and placement constraints $P : A \to 2^R$:*

Decision: *Does there exist assignment $\sigma : A \to R$ such that:*

1) *$\sigma(a_i) \in P(a_i)$ for all a_i (placement constraints)*
2) *$(a_i, a_j) \in C \Rightarrow \sigma(a_i) \neq \sigma(a_j)$ (no conflicts)*

Optimization: *Minimize $|\{\sigma(a_i) : a_i \in A\}|$ (fewest rooms).*

This generalizes Graph k-Coloring with per-vertex color restrictions (list coloring).

C. Reduction from Graph k-Coloring

Theorem 3 (NP-Completeness)**.** *Museum Artwork Arrangement (decision version) is NP-Complete.*

Proof. **(In NP)** Given assignment $\sigma : A \to R$, verify in polynomial time:

- Placement constraints: $O(n)$ checks
- Conflict constraints: $O(|C|)$ checks

Total: $O(n + |C|) = O(n^2)$. Thus Museum Arrangement $\in$ NP.

(NP-Hardness via Reduction from Graph k-Coloring) Graph k-Coloring is NP-Complete [2]: Given graph $G = (V, E)$ and integer k, does a proper k-coloring exist?

Construction (polynomial time):

- For each vertex $v_i \in V$, create artwork $a_i \in A$
- For each edge $(v_i, v_j) \in E$, add conflict $(a_i, a_j) \in C$
- Set rooms $R = \{r_1, \ldots, r_k\}$ (exactly k rooms)
- Set all placement constraints: $P(a_i) = R$ for all a_i

Construction time: $O(|V| + |E|)$.

Correctness ($\Rightarrow$): Suppose G has valid k-coloring $\chi : V \to \{1, \ldots, k\}$. Define $\sigma(a_i) = r_{\chi(v_i)}$. Since χ is proper, adjacent vertices have different colors, so conflicting artworks have different rooms. All $P(a_i) = R$, so placement constraints hold. Thus σ is valid.

Correctness ($\Leftarrow$): Suppose Museum Arrangement has valid $\sigma : A \to R$. Define $\chi(v_i) = j$ where $\sigma(a_i) = r_j$. Since no conflicting artworks share rooms, no adjacent vertices share colors. Thus χ is a valid k-coloring.

By bidirectional correspondence, Graph k-Coloring $\leq_p$ Museum Arrangement. Since Graph k-Coloring is NP-Complete, Museum Arrangement is NP-Hard. Combined with NP membership, Museum Arrangement is NP-Complete. □

D. Greedy Approximation Algorithms

Exact solution requires exponential time (unless P=NP). We employ two greedy heuristics:

1) Welsh-Powell Algorithm: Places artworks by decreasing conflict degree, assigning first available room from allowed set.

Algorithm 3 Welsh-Powell Museum Arrangement

```
Require: Artworks $A$, conflicts $C$, allowed rooms $P$
Ensure: Assignment $\sigma : A \to R$ or INFEASIBLE
1: Sort $A$ by $|C_a|$ descending where $C_a = \{a' : (a, a') \in C\}$
2: for each artwork $a$ in sorted order do
3:     $used \leftarrow \{\sigma(a') : a' \in C_a, a'$ assigned$\}$
4:     $available \leftarrow P(a) \setminus used$
5:     if $available = \emptyset$ then
6:         return INFEASIBLE
7:     else
8:         $\sigma(a) \leftarrow \min(available)$
9: return $\sigma$
```

2) DSatur Algorithm: Dynamically selects artwork with highest saturation degree (distinct rooms in conflicting neighbors).

Algorithm 4 DSatur Museum Arrangement

```
Require: Artworks $A$, conflicts $C$, allowed rooms $P$
Ensure: Assignment $\sigma$ or INFEASIBLE
1: $unplaced \leftarrow A$
2: while $unplaced \neq \emptyset$ do
3:     Find $a^* \in unplaced$ with max:
4:         1) $sat(a) = |\{\sigma(a') : a' \in C_a, a'$ placed$\}|$
5:         2) $deg(a) = |\{a' \in C_a : a' \in unplaced\}|$ (tiebreak)
6:     $used \leftarrow \{\sigma(a') : a' \in C_{a^*}, a'$ placed$\}$
7:     $available \leftarrow P(a^*) \setminus used$
8:     if $available = \emptyset$ then
9:         return INFEASIBLE
10:    else
11:        $\sigma(a^*) \leftarrow \min(available)$; $unplaced \leftarrow unplaced \setminus \{a^*\}$
12: return $\sigma$
```

Theorem 4 (Greedy Heuristic Complexity)**.** *Welsh-Powell and DSatur run in $O(n^2)$ time where $n = |A|$ and both terminate with valid colorings (if feasible) or correctly report infeasibility.*

Proof. **Welsh-Powell Termination and Feasibility:** The algorithm processes each artwork exactly once in sorted order. For each artwork a, it examines at most $|P(a)| \leq |R|$ rooms and at most $|C_a| \leq n$ conflicting neighbors. Since the list is finite and each artwork is processed once, the algorithm terminates in finite time.

At each step, the algorithm assigns artwork a to a room $r \in P(a)$ such that no already-colored neighbor $a' \in C_a$ occupies r. This maintains the invariant: *the partial coloring is valid*. If no such room exists, the algorithm correctly reports infeasibility (no valid coloring exists for the given conflict graph and placement constraints under greedy ordering). By induction on the number of colored artworks, the final coloring (if produced) satisfies all constraints.

Welsh-Powell Complexity: Sorting by conflict degree takes $O(n \log n)$. For each artwork, checking neighbor rooms requires iterating over conflicts ($O(\deg(a))$), bounded by $O(n)$ per artwork. Total: $O(n \log n + n \cdot n) = O(n^2)$.

DSatur Termination and Feasibility: The algorithm maintains a set of uncolored artworks, which decreases by exactly one per iteration. Since $|A|$ is finite, the algorithm terminates in at most n iterations.

Each iteration selects the uncolored artwork a^* with maximum saturation degree (tiebroken by uncolored degree) and assigns it a room $r \in P(a^*)$ not used by any colored neighbor. This maintains the invariant: *the partial coloring respects all conflict constraints*. If no valid room exists for a^*, the algorithm correctly reports infeasibility. By induction, the final coloring satisfies all constraints.

DSatur Complexity: Each iteration selects one artwork (finding maximum saturation requires scanning all uncolored artworks: $O(n)$ per iteration, plus $O(n)$ to compute saturation for each candidate by checking neighbors), then checks neighbor rooms ($O(n)$). With n iterations, total time is $O(n \cdot (n + n)) = O(n^2)$.

Approximation Quality: Graph coloring admits no polynomial-time approximation within $n^{1-\epsilon}$ unless P=NP [15]. Thus greedy heuristics provide no provable approximation guarantee for the general problem. However, they perform well on sparse graphs and are standard practice for graph coloring due to their simplicity and speed. □

E. Experimental Validation

Implemented in Python 3.11 on same hardware as Problem A. Small instances ($n \leq 15$) verified valid arrangements (no violations) across 20 trials. For scalability, we generated conflict graphs with edge probability 0.07 and measured runtime/quality. Table II shows results.

TABLE II
MUSEUM ARRANGEMENT HEURISTICS PERFORMANCE

n	Conflicts	Welsh-Powell		DSatur	
		Time (s)	Rooms	Time (s)	Rooms
10	2	0.00003	2	0.00011	2
20	8	0.00008	4	0.00047	4
30	30	0.00012	3	0.00090	4
40	60	0.00011	5	0.00119	5
50	97	0.00016	5	0.00200	4
60	138	0.00017	4	0.00271	5

Both heuristics exhibit $O(n^2)$ scaling (Fig. 2). DSatur produces competitive or better colorings (Fig. 3), confirming theoretical advantage [9].

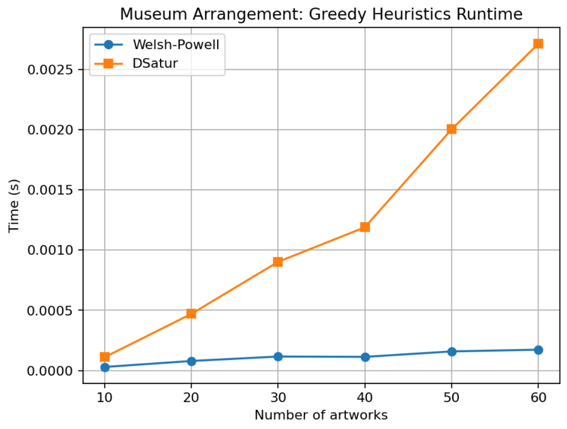

Fig. 2. Runtime comparison showing quadratic scaling.

IV. COMPARISON AND DISCUSSION

Table III summarizes fundamental differences.

Key insight: structural properties (flow conservation vs. conflict constraints) determine tractability [2]. Network flow benefits from linearity enabling polynomial algorithms [6], while graph coloring requires global consistency checks resisting efficient solution [15].

Practically, P vs. NP matters: blood distribution computes provably optimal allocations in milliseconds, while museum curation requires heuristics potentially missing better arrangements.

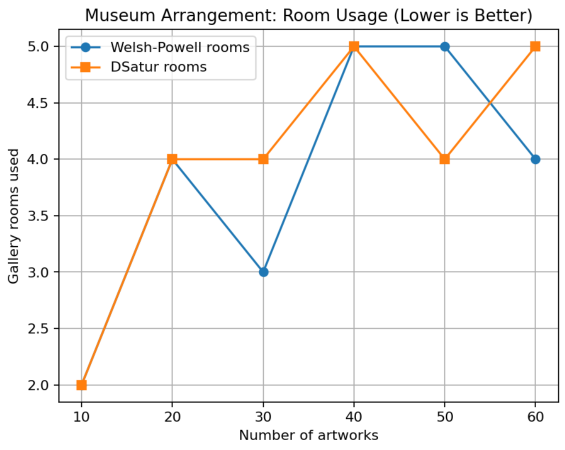

Fig. 3. Gallery rooms used (lower is better).

TABLE III
COMPLEXITY CLASS COMPARISON

Property	Blood Dist.	Museum Arr.
Complexity	P	NP-Complete
Reduction	Max Flow	Graph Coloring
Algorithm	Edmonds-Karp	Greedy Heuristics
Optimality	Guaranteed	Approximation
Time	$O(VE^2)$	$O(n^2)$ (greedy)
Exact Solution	Polynomial	Exponential

V. CONCLUSION

We demonstrated two problems from different complexity classes: blood distribution (polynomial via network flow) and museum arrangement (NP-Complete via graph coloring). Through formal reductions, correctness proofs, and empirical validation, we showed how complexity theory guides algorithm design.

For blood distribution, Edmonds-Karp guarantees optimal solutions polynomially. For museum arrangement, NP-Completeness necessitates greedy approximations running efficiently but sacrificing optimality.

These results underscore a central tenet: problem structure—not just size—determines tractability [2].

SOURCE CODE REPOSITORY

Complete implementation available at:
https://github.com/AbhinavLakkapragada/AOA-Project-2

LLM AND TOOLING DISCLOSURE

We used OpenAI's ChatGPT (GPT-4o) and Anthropic's Claude (Sonnet 4.5) for LaTeX formatting and prose refinement. All algorithmic content, reductions, proofs, and experiments were human-authored and independently verified.

Example Interactions

Prompt 1: "Help structure a formal reduction from Graph k-Coloring to museum artwork placement with room constraints."

Response (excerpt): "Map each vertex to an artwork, each edge to a conflict, set k rooms, allow all artworks in any room..."

How Used: Adapted construction but added placement constraint handling ($P(a_i) = R$ for reduction vs. $P(a_i) \subseteq R$ generally). Expanded bidirectional proof with explicit correctness.

Prompt 2: "Explain multi-layer network flow construction for blood distribution with compatibility."

Response (excerpt): "Create layers: source $\rightarrow$ banks $\rightarrow$ blood types $\rightarrow$ hospitals $\rightarrow$ sink with compatibility edges..."

How Used: Adopted structure but formalized capacity definitions, proved correctness via Max-Flow Min-Cut, developed experimental validation independently.

Other interactions:

- "Pseudocode for Edmonds-Karp with BFS augmenting paths."
- "Welsh-Powell and DSatur implementation in Python."
- "Experimental timing harnesses with matplotlib plots."
- "IEEE template formatting with theorem environments."

All proofs, complexity analysis, code, and results were independently authored.

Appendix

```
def edmonds_karp(self, source, sink):
    parent = {}; max_flow = 0
    residual = defaultdict(lambda: defaultdict(int))
    for u in self.graph:
        for v in self.graph[u]:
            residual[u][v] = self.graph[u][v]

    while self.bfs_find_path(source, sink, parent, residual):
        path_flow = float('inf'); v = sink
        while v != source:
            u = parent[v]
            path_flow = min(path_flow, residual[u][v])
            v = u

        v = sink
        while v != source:
            u = parent[v]
            residual[u][v] -= path_flow
            residual[v][u] += path_flow
            v = u
        max_flow += path_flow; parent.clear()
    return max_flow
```

Listing 1. Edmonds-Karp Implementation (Excerpt)

```
def greedy_welsh_powell(artworks, conflicts):
    graph = build_conflict_graph(artworks, conflicts)
    degrees = [(len(graph[a.id]), a) for a in artworks]
    degrees.sort(reverse=True)
    assignment = {}

    for _, artwork in degrees:
        neighbor_rooms = {assignment[n] for n in graph[artwork.
              id]
                         if n in assignment}
        assigned = False
        for room in GALLERY_ROOMS:
            if room in artwork.allowed_rooms and room not in
                  neighbor_rooms:
                assignment[artwork.id] = room
                assigned = True; break
        if not assigned: return None
    return assignment
```

Listing 2. Welsh-Powell Heuristic (Excerpt)

Full code available at the GitHub repository listed above.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th ed. MIT Press, 2022.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to NP-Completeness*. Freeman, 1979.

[3] M. Sipser, *Introduction to the Theory of Computation*, 3rd ed. Cengage, 2012.

[4] S. A. Cook, "The complexity of theorem-proving procedures," *STOC*, 1971, pp. 151–158.

[5] V. V. Vazirani, *Approximation Algorithms*. Springer, 2001.

[6] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian J. Math.*, vol. 8, pp. 399–404, 1956.

[7] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow," *J. ACM*, vol. 19, no. 2, pp. 248–264, 1972.

[8] D. J. A. Welsh and M. B. Powell, "An upper bound for the chromatic number," *Comput. J.*, vol. 10, no. 1, pp. 85–86, 1967.

[9] D. Brélaz, "New methods to color vertices," *Commun. ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[10] K. E. Kendall, "Blood bank decision making," *Med. Decis. Making*, vol. 1, no. 1, pp. 73–90, 1981.

[11] American Red Cross, "Blood Types," 2024. [Online]. Available: https://www.redcross.org/blood

[12] B. Lord and G. D. Lord, *Museum Planning*, 3rd ed. Altamira, 2009.

[13] D. Saunders and J. H. Kirby, "Effect of humidity on pigments," *Nat. Gallery Tech. Bull.*, vol. 25, pp. 62–72, 2004.

[14] J. H. Falk and L. D. Dierking, *The Museum Experience Revisited*. Left Coast, 2013.

[15] S. Khanna, N. Linial, and S. Safra, "Hardness of chromatic number," *Combinatorica*, vol. 20, no. 3, pp. 393–415, 2000.