Predictive Flows for Faster Ford-Fulkerson

Sami Davies [*1] **Benjamin Moseley** [*2] **Sergei Vassilvitskii** [*3] **Yuyan Wang** [*3]

Abstract

Recent work has shown that leveraging learned predictions can improve the running time of algorithms for bipartite matching and similar combinatorial problems. In this work, we build on this idea to improve the performance of the widely used Ford-Fulkerson algorithm for computing maximum flows by seeding Ford-Fulkerson with predicted flows. Our proposed method offers strong theoretical performance in terms of the quality of the prediction. We then consider image segmentation, a common use-case of flows in computer vision, and complement our theoretical analysis with strong empirical results.

1. Introduction

The Ford-Fulkerson method is one of the most ubiquitous in combinatorial optimization, both in theory and in practice. While it was first developed for solving the maximum flow problem, many problems in scheduling (Ahuja et al., 1993), computer vision (Vineet and Narayanan, 2008), resource allocation, matroid intersection (Im et al., 2021b), and other areas are solvable by finding a reduction to a flow problem.

Theoretically, max flow algorithms exist that are asymptotically much faster than the original Ford-Fulkerson formulation, most recently the near-linear time algorithm of Chen et al. (2022b). However—as often happens—algorithms with great theoretical guarantees might be difficult to implement in practice. Indeed, algorithms used in practice still leave room for improvement. In fact, for computing max flows in networks, practitioners often stick to older algorithms such as Dinic's algorithm (Bhadra et al., 2020), push-relabel (Cherkassky and Goldberg, 1997), pseudoflow (Chandran and Hochbaum, 2009), or these algorithms layered with heuristics to fit particular settings.

[*]Equal contribution [1]Department of Computer Science, Northwestern University, Evanston IL, USA [2]Tepper School of Business, Carnegie Mellon University, Pittsburgh PA, USA [3]Google Research, NYC NY, USA. Correspondence to: Yuyan Wang <wangyy@google.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

When flow algorithms are deployed in practice, they are often used to solve several problem instances arising naturally over time. However, the theoretical analysis, as well as many implementations, considers solving each new problem from scratch to derive worst-case guarantees. This approach needlessly discards information that may exist between instances. We are interested in discovering whether flow problems can be solved more efficiently by leveraging information from past examples. Seeding an algorithm from a non-trivial starting point is referred to as a **warm-start**.

We are motivated by the question: can one warm-start Ford-Fulkerson to improve theoretical and empirical performance? Towards this goal, we leverage the recently developed algorithms with predictions framework (a.k.a learning-augmented algorithms). Research over the past several years has showcased the power of augmenting an algorithm with a learned prediction, leading to improvements in caching (Im et al., 2022; Lindermayr and Megow, 2022; Lykouris and Vassilvitskii, 2021), scheduling (Im et al., 2021a; Lattanzi et al., 2020), clustering (Lattanzi et al., 2021), matching (Chen et al., 2022a; Dinitz et al., 2021), and more (see the survey by Mitzenmacher and Vassilvitskii (2022)). An algorithm is **learning-augmented** if it can use a prediction that relays information about the problem instance. Most prior work uses predictions to overcome uncertainty in the online setting. However, recent work by Dinitz et al. (2021)—and the follow-up work by Chen et al. (2022a)—instead focuses on improving the run-time of bipartite matching algorithms by predicting the dual variables and using these to warm-start the primal-dual algorithm.

Motivated by the idea of warm-starting combinatorial optimization algorithms, we seek to provide faster run-time guarantees for flow problems via warm-start. The paper will focus on flow problems generally, but will additionally showcase a common, practical use-case in computer vision: image segmentation. In the image segmentation problem, the input is an image containing an object/ foreground, and the goal is to locate the foreground in the image.

1.1. Our contributions

For a graph $G = (V, E)$ equipped with a capacity vector $c \in \mathbb{Z}_{\geq 0}^{|E|}$, let f be a flow on G, where f_e is the flow value on each edge $e \in E$. Let $\mathcal{F}^*$ be the collection of all feasible,

maximum flows on G. Given a potentially infeasible flow $\widehat{f}$, let $\eta(\widehat{f}) = \min_{f^* \in \mathcal{F}^*} ||\widehat{f} - f^*||_1$. This term denotes how close $\widehat{f}$ is to being optimal.

Warm-starting Ford-Fulkerson on general networks Our main contribution is Algorithm 1, which can be used to warm-start any implementation of Ford-Fulkerson, i.e., Ford-Fulkerson with any specified subroutine for finding augmenting paths. Algorithm 1 takes as input a predicted flow $\widehat{f}$. Note $\widehat{f}$ may be infeasible for G, as predictions can be erroneous. Algorithm 1 first projects $\widehat{f}$ to a feasible flow for G, and then runs the Ford-Fulkerson procedure from the feasible state to find a maximum flow. While our warm-started Ford-Fulkerson has its performance tied to the quality of the prediction, it also enjoys the same worst-case run-time bounds as the vanilla Ford-Fulkerson procedure.

Theorem 1. *Let $\widehat{f}$ be a potentially infeasible flow on network $G = (V, E)$. Let T be the worst-case run-time for Ford-Fulkerson with a chosen augmenting path subroutine. Using the same subroutine, Algorithm 1 seeded with $\widehat{f}$ finds an optimal flow f^* on G within time $O(\min\{|E| \cdot \eta(\widehat{f}), T\})$*[1].

At various points, we specify two Ford-Fulkerson implementations: Edmonds-Karp and Dinic's algorithm, for which the run-time T is $O(|E|^2|V|)$ and $O(|V|^2|E|)$, respectively.

One may wonder how to obtain such a $\widehat{f}$. While in practice it might be sufficient to use, for warm-start, the optimal solution found on a past problem instance, recent studies have proposed the PAC-learning framework for finding the "best" overall prediction. To this end, we prove that when the networks come from a fixed but unknown distribution, one can indeed PAC-learn the best approximation (Theorem 9) for optimal flows.

Faster warm-start on locally-changed networks Next, we improve the analysis of Algorithm 1 for network instances with gradual, local transitions from one to another. We prove the local transitions among networks, informally characterized in Theorem 2, give rise to many short paths along which we can send flow, thus improving the run-time.

Theorem 2 (Informal, formally Theorem 10)**.** *Fix separable networks G^1 and G^2, where the transition between them is d-local. For $\widehat{f}$ an optimal flow on G^1, the run-time of Algorithm 1 seeded with $\widehat{f}$ on G^2 to find optimal f^* on G_2 is $O(d^2 \cdot |E| + d \cdot \eta(\widehat{f}))$.*

Empirical results Motivated by our theoretical results, we use our warm-started Ford-Fulkerson procedure on networks derived from instances of *image segmentation* on sequences of photos taken of a moving object or from changing angles. We show that warm-start is faster than standard Ford-Fulkerson procedures (2-5$\times$ running time improvements), thus demonstrating that our theory is predictive of practical performance. A key piece of the speed gain of warm-start comes not from sending the flow along fewer paths, but rather from using shorter paths to project $\widehat{f}$ to a feasible flow, as predicted by Theorem 2. We note that the goal of our experiments is not necessarily to provide state-of-the-art algorithms for image segmentation, but instead to show that warm-starting Ford-Fulkerson leads to substantial run-time improvements on practical networks as compared to running Ford-Fulkerson from scratch.

[1]Here the term $O|E|$ generally refers to the run time of finding each augmenting path since the most commonly used subroutines are BFS/DFS. However, given any other subroutine with different time bounds one can also replace this term.

1.2. Related work

Flow problems have been well studied. See the survey by Ahuja et al. (1993). The Ford-Fulkerson method greedily computes a maximum flow by iteratively using an augmenting-path finding subroutine (Ford and Fulkerson, 1956). Different subroutines give rise to different implementations such as Edmonds-Karp (using BFS) (Edmonds and Karp, 1972) and the even faster Dinic's algorithm (Dinitz, 2006). Sherman (2013) and Kelner et al. (2014) give fast algorithms that compute approximate maximum flows. The theoretical breakthrough by Chen et al. (2022b) gave a nearly-linear time max flow algorithm, and we note that the current presentation of this algorithm is not appropriate for implementation; huge steps are still needed in order to make this algorithm that is actually usable for practitioners.

Similar in spirit to our work, Altner and Ergun (2008) demonstrate empirically that one can warm-start the push-relabel algorithm on similar networks. Additionally, we are aware of concurrent work on a warm-started max flow algorithm by Polak and Zub (2022). Importantly, they require an additional assumption that the predicted flow satisfy flow conservation constraints, a limitation that the authors highlight. In contrast, we have an explicit feasibility restoration step, allowing us to get rid of this assumption.

Learning-augmented algorithms have become popular recently. The area was jump started by Kraska et al. (2018), who showed results on learned data structures. The area has since become popular for the design of online algorithms where the algorithm uses predictions to cope with uncertainty (Purohit et al., 2018; Lattanzi et al., 2020; Antoniadis et al., 2020). See also the paragraph *Learning-augmented algorithms* in Section 1.3.

While Kraska et al. (2018) showed that running times can be improved using predictions, this is still yet to be well-understood theoretically. The work of Dinitz et al. (2021) showed how to improve the run-time of the Hungarian algorithm for weighted bipartite matching. Chen et al. (2022a) has extended this to other graph problems. Both of these

works warm-start primal dual algorithms with a predicted dual solution, and it is reasonable to wonder whether those techniques can be extended to the general flow problem considered in our paper. We do not believe this is possible. The work of Chen et al. (2022a) could only potentially handle the restricted case of 0-1 min-cost flow (i.e., the capacities of the edges in the graph are 0 or 1) by reducing that problem to min-weight bipartite perfect matchings and then using their algorithm. This is because several aspects of their run-time analysis heavily rely on the assumption that the constraints are 0-1. Moreover, the reductions to bipartite matching that allowed Chen et al. (2022a) to study other combinatorial optimization problems (including 0-1 min-cost flow) do not exist for general integral flow problems.

Other run-time improvements have been made using predictions, too. A more general framework for warm-starting algorithms is given by Sakaue and Oki (2022), though we note that their method does not work for general optimization problems (only L-convex ones), and therefore cannot be used on general flow problems. Additionally, Lu et al. (2021) showed predictions can be used to improve the run-time of generalized sorting. A closely related area is that of data-driven algorithm design (Gupta and Roughgarden, 2017; Balcan et al., 2021).

1.3. Organization and preliminaries

Section 2 presents the warm-start algorithm, proves its correctness, and provides run-time guarantees. In Section 3, we give additional theoretical guarantees if the networks are from a specific subclass. In Section 4, we show our empirical results on networks arising from image segmentation, which are closely related to the networks in Section 3.

Learning-augmented algorithms In this model, an algorithm is given access to a predicted parameter. The prediction can be erroneous and must come equipped with an error metric. In our setting, for a given network G, we will predict a flow $\widehat{f}$. This predicted flow may be infeasible for G. Recall that for $\mathcal{F}^*$ the set of optimal flows on G, we define the error of $\widehat{f}$ on G to be $\eta(\widehat{f}) = \min_{f^* \in \mathcal{F}^*} ||\widehat{f} - f^*||_1$.

It is well-established in the literature (see, for example, (Lykouris and Vassilvitskii, 2021)) on learning-augmented algorithms that the desired properties of the prediction and algorithm are *learnability, consistency, and robustness*. We show that given an additional assumption on the uniqueness of optimal flows (see Assumption 8), predicted flows are PAC-learnable in Theorem 9. Additionally, we use the **instance-robustness** (Lavastida et al. (2021)) of flows to justify learnability for special networks in Theorem 2 and observe this empirically, as well.

If we are given a predicted flow $\widehat{f}$ for a network G that is actually an optimal flow, then the run-time is 0, so Algorithm 1 is consistent. We are also guaranteed robustness and a worst-case guarantee, as the run-time of Algorithm 1 is $O(\min\{|E|\eta(\widehat{f}), T\})$, which degrades smoothly when $\widehat{f}$ is far from an optimal flow f^* on a network, but the worst-case is still bounded by $O(T)$[2]. We note that the run-time bounds in learning-augmented algorithms are often (if not always) asymptotically the same as the worst-case bound of the vanilla algorithm, but typically with a worse leading constant. Such constants represent the overhead of trying to make use of a (potentially wrong) prediction. For example, this constant is 2 in the caching algorithm in Lykouris and Vassilvitskii (2021) (and our constant is also 2).

Network flow Let $G = (V, E)$ be a fixed network with $|V| = n$ and $|E| = m$. The source s and sink t are part of the vertex set V. The network is equipped with a capacity vector $c \in \mathbb{Z}^m_{\geq 0}$. A flow $f \in \mathbb{Z}^m_{\geq 0}$ on G is feasible if it satisfies flow conservation, i.e. for all vertices $u \in V \setminus \{s, t\}$ the incoming flow for u equals the outgoing flow $\sum_{e=(v,u)} f_e = \sum_{e=(u,w)} f_e$, and capacity constraints, i.e. $f_e \leq c_e$ for all edges $e \in E$. Throughout the paper we refer to a flow satisfying these constraints as **feasible**.

Given a flow f that satisfies capacity constraints but not necessarily flow conservation, the **residual graph** G_f is the network on the same set of vertices, and for any edge $e = (u, v) \in E$, add edge e to G_f but with capacity $c'_e = c_e - f_e$ and a reversed edge $\overleftarrow{e} = (v, u)$ with capacity f_e. Let $\nu(f)$ be the amount of flow that f sends from the source, $\nu(f) = \sum_{e=(s,u)} f_e$. An **augmenting path** in G_f is a path p from s to t where every edge $e \in p$ has $c'_e > 0$.

When f does not satisfy flow conservation, the total in- and out-coming flow values on a node are different. Call this difference the **excess/deficit** ($\mathsf{ex}_f/\mathsf{def}_f$) of the node if in-coming is more/less than out-coming flows respectively. For shorthand, we let the total excess and deficit in G according to flow f be $\mathsf{ex}_f = \sum_{u \notin \{s,t\}} \mathsf{ex}_f(u)$ and $\mathsf{def}_f = \sum_{u \notin \{s,t\}} \mathsf{def}_f(u)$; note that this excludes the source and sink, which are special nodes with deficit/excess by definition. Let $A_f, B_f \subseteq V$ be the nodes with positive excess/deficit with respect to f (so $t \in A_f$, $s \in B_f$), respectively. It will be convenient to further refer to these sets **excluding** s, t with $A'_f = A_f \setminus \{t\}$ and $B'_f = B_f \setminus \{s\}$.

2. Warm-start Ford-Fulkerson

Here, we give our algorithm for using predicted flows. The next proposition follows from the following observations. Any Ford-Fulkerson method (e.g., Edmonds-Karp or Dinic's) can be seeded with any feasible flow. Each iteration of Ford-Fulkerson increases the value of the flow and takes $O(|E|)$ time to find an augmenting path and send flow.

[2]Recall, T depends on the Ford-Fulkerson implementation.

Proposition 3. *Let f be a feasible flow on G, where $\nu(f) < \nu(f^*)$ for f^* an optimal flow on G. Ford-Fulkerson seeded with f terminates in at most $\nu(f^*) - \nu(f)$ many iterations, so its run-time is $O(|E| \cdot (\nu(f^*) - \nu(f)))$.*

Let $\widehat{f}$ be a predicted flow for network G. It may be infeasible, that is, it can violate capacity or flow conservation constraints. Algorithm 1 has two primary steps: step (1) projects $\widehat{f}$ to a feasible flow—we call this the **feasibility projection**—and step (2) runs a Ford-Fulkerson method seeded with a feasible flow and finds an optimal flow. During feasibility projection, we first round down the flow wherever capacity constraints are violated. Then we send flow along **projection paths**, that is, a path from excess nodes to deficit nodes where all capacities are positive in the residual graph, to get flow conservation.

Algorithm 1 Warm-starting Ford-Fulkerson with $\widehat{f}$

Input: predicted flow $\widehat{f}$
while $\exists$ edge e in G with $\widehat{f}_e > c_e$ **do**
  Update $\widehat{f}$ to $\widehat{f}_e \leftarrow c_e$
end while
Set $f \leftarrow \widehat{f}$
Build the residual graph G_f, as well as A'_f and B'_f, the sets of nodes with excess/deficit to round
// Main while loop, feasibility projection
while $|A'_f \cup B'_f| > 0$ **do**
  if $|A'_f| > 0$ **then**
    if $\exists$ projection path from $u \in A'_f$ to $v \in B'_f$ **then**
      Let p be the path from u to v
    else
      Let p be a path from $u \in A'_f$ to s
    end if
  else
    Let $v \in B'_f$, let p be a path from t to v
  // Find projection paths by adapting the augmenting path subroutine. See the text for more details.
  end if
  For w, w' beginning, ending nodes of p respectively
  Find flow $\mu_p = \min\{\mathsf{ex}_f(w), \mathsf{def}_f(w'), \min_{e \in p} c'_e\}$
  Send μ_p units down path p, Update f, G_f, A'_f, and B'_f
end while
// Feasibility projection ends, optimization starts
Run Ford-Fulkerson on G seeded with f until optimality
Output: f^*

In the main WHILE loop of Algorithm 1, projection paths are found in three rounds: $A'_f - B'_f$, $A'_f - s$, $t - B'_f$. Within each round, we find the projection paths by constructing auxiliary graphs G' and applying on this graph the chosen augmenting path subroutine (e.g., BFS) in Ford-Fulkerson. To build G' for finding all possible $A'_f - B'_f$ projection paths, take the residual graph G_f w.r.t f and treat it as a new network. Add a super source s^* and super sink t^*. Add arcs (s^*, u) to every $u \in A'_f$ (non-sink excess nodes) with capacity $\mathsf{ex}_f(u)$, and (u, t^*) from every $u \in B'_f$ (non-source deficit nodes) to t^*. Initialize all flows to be 0 on this network. An $s^* - t^*$ augmenting path in G' corresponds to a $A'_f - B'_f$ projection path in G_f. This is because for any projection path from u to u' one can let there be flow on s^* to u and u' to t^* and thus making it an augmenting path in G', and the reverse procedure holds. The graph G' for the other two rounds are built similarly; for finding $A'_f - s$ projection paths, add arcs (s^*, u) to $u \in A'_f$ and (s, t^*); for $t - B'_f$, add arcs (s^*, t) and (v, t^*) to every $v \in B'_f$.

2.1. Warm-start Algorithm Analysis

In this section we analyze how Algorithm 1 works.

Validity of algorithm We first prove that the projection paths can be found following the order of $A'_f - B'_f$, then $A'_f - s$, and lastly $t - B'_f$.

Lemma 4. *Fix an infeasible flow f. If there is no path from A'_f to B'_f with positive capacity in G_f, then sending flow from A'_f to s (or from t to B'_f) to form the flow h will not result in a path from A'_h to B'_h with positive capacity in G_h.*

The proof can be found in Appendix 6.1. We next prove that the projection path must always exist in the main WHILE loop as long as $|A'_f \cup B'_f|$ is not empty.

Lemma 5. *Given infeasible flow f, $\forall u \in A'_f$, $\exists v \in B_f$ such that there is a projection path from u to v; $\forall v \in B'_f$, $\exists u \in A_f$ such that there is a projection path from u to v.*

This lemma results from the following observation that links the summation of excess/deficit to the difference between in-flows and out-flows for any fixed set of nodes.

Proposition 6. *Let f be a flow satisfying the capacity constraints of a network G. Then for any $S \subseteq V$, the difference between the total deficits in S and the total excesses in S is exactly the difference between the total flow out of S and into S. Formally,*

$$\sum_{u \in S} \mathsf{def}_f(u) - \sum_{u \in S} \mathsf{ex}_f(u) = \sum_{u \in S} \sum_{\substack{e=(u,v): \\ v \notin S}} f_e - \sum_{u \in S} \sum_{\substack{e=(v,u): \\ v \notin S}} f_e.$$

Proof. In $\sum_{u \in S} \mathsf{def}_f(u) - \sum_{u \in S} \mathsf{ex}_f(u)$, the edges with flow within S are counted with a positive and negative sign, but the edges carrying flow into S or out of S are counted once by the excess and once by the deficit, respectively. □

Proof of [Lemma 5] We prove one direction by contradiction. The proof for the other direction is similar. For any $u \in A'_f$, assume no such path exists. In Proposition 6 take S to be the set of all vertices reachable from u in G_f. None of the nodes in S can have positive deficit, so

the LHS of Proposition 6 must be negative. On the other hand, S must have 0 flow incoming to it, otherwise there is an edge pointing from S to $V \setminus S$ in G_f, producing a vertex in $V \setminus S$ reachable from u and contradicting the maximality of S. Therefore, the RHS in Proposition 6 is non-negative, contradicting the equation. □

Note each iteration decreases the total amount of excess and deficit in the system, $\mathsf{ex}_f + \mathsf{def}_f$, by at least one, so the WHILE loop terminates after restoring flow conservation, giving rise to a feasible flow. Then the vanilla Ford-Fulkerson takes over until an optimal solution is found.

Running-time analysis Here we prove Theorem 1. It is straight-forward to justify the run-time of Algorithm 1 is bounded by $O(T)$. During the feasibility projection step an auxiliary graph is constructed three times, each time with $|V| + 2$ vertices and $O(|E|)$ edges. Thus running the chosen Ford-Fulkerson implementation on these graphs takes time $O(T)$. The optimization step also takes time $O(T)$, since running Ford-Fulkerson starting with a feasible flow is equivalent to running it from scratch on the residual graph as a new input. Thus the total running time is $O(T)$.

On the other hand, we show the running time is also tied to the quality of prediction, $\eta(\widehat{f}) = \min_{f^* \in \mathcal{F}^*} ||\widehat{f} - f^*||_1$. We note that in case of multiple max flow solutions, $\eta(\cdot)$ is evaluated at the f^* closest to the seed solution $\widehat{f}$, which does not necessarily coincide with the max flow solution returned by Algorithm 1. We first bound the times the path-finding subroutine is called. For projection paths, the total excess and deficit could increase by at most $\sum_e \max\{\widehat{f}_e - c_e, 0\}$ when Algorithm 1 rounds down the flow where it exceeds capacity. Thus it takes at most $(\mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}})$ projection paths to restore feasibility. For augmenting paths, the difference in flow value, $\nu(\widehat{f}) - \nu(f^*)$, could decrease by at most $\sum_e \max\{\widehat{f}_e - c_e, 0\}$ during the round-down and another $\mathsf{ex}_{\widehat{f}}$ during feasibility projection, thus the total number is at most the summation of the three. Each path-finding takes $O(|E|)$ time. This combined with the next lemma proves Theorem 1; the full proof is deferred to Appendix 6.1.

Lemma 7. *$\nu(f^*) - \nu(f)$, $\mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}}$, and $\sum_e \max\{\widehat{f}_e - c_e, 0\}$ are upper bounded by $\eta(\widehat{f})$.*

2.2. PAC-learning Flows

Here, we show theoretically that high quality flows are learnable. This gives theoretical evidence that flows can be learned for input to Algorithm 1. We show that given a distribution over capacity vectors for a network, one can learn a predicted flow from samples that is the best approximation. Importantly, PAC learning says nothing about the relative quality of the solution, other than saying it is the best possible

Generally speaking, while experiments give empirical evidence that predicting flows is reasonable, PAC-learning results give the theoretical evidence. The learning method presented in this section, where one takes the predicted flow to be the median of sampled flows on each edge, is a standard learning method in the related literature (see either of the faster matchings works by Dinitz et al. (2021) or Chen et al. (2022b)).

Consider a fixed network G with edge capacities c. An instance is a network G^i on the same vertex and edge set as G, but the capacity vector is c^i, where every edge e in G^i must satisfy $c^i_e \in [0, c_e]$. Let $\mathcal{D}$ be an unknown distribution over such instances. Since an instance is exactly characterized by its new capacity vector, we notationally write this as sampling a capacity vector $c^i \sim \mathcal{D}$.

Suppose we sample instances $c^1, \ldots, c^s$ from $\mathcal{D}$. Let $\mathcal{F}$ be the set of all integral flows on G that satisfy the capacities in c, noting that flows in $\mathcal{F}$ do not have to satisfy flow conservation. Technically, a network G might have several optimal solutions. Here we make the following assumption.

Assumption 8. *For a network G, there is a uniquely associated, computable optimal flow.*[3]

Given samples $c^1, \ldots, c^s$, we can compute the uniquely associated optimal flows on those samples $f^*(c^1), \ldots, f^*(c^s)$. One can efficiently compute the empirical risk minimizer in $\widehat{f} \in \mathcal{F}$ to be the coordinate-wise median of the optimal flows of the samples, i.e. $\widehat{f}_e = \mathsf{median}(f^*(c^1)_e, \ldots, f^*(c^s)_e)$ for all $e \in E$ (see Lemma 11 in Appendix 6.2).

We will now state our PAC-learning result, whose proof is in Appendix 6.2. The proof of this theorem follows that of (Dinitz et al., 2021).

Theorem 9. *Let $c^1, \ldots, c^s$ be sampled i.i.d. from $\mathcal{D}$ and let $\widehat{f} = \mathit{argmin}_{f \in \mathcal{F}_0} \frac{1}{s} \sum_{j=1}^{s} ||f - f^*(c_j)||_1$. For*

$$s = \Omega((\max_e c_e^2 \cdot m^2)(\log m + \log(1/p)))$$

and $\widetilde{f} = \mathit{argmin}_{f \in \mathcal{F}_0} \mathbb{E}_{c^i \sim \mathcal{D}} ||f - f^*(c^i)||_1$, then with probability at least $1 - p$, $\widehat{f}$ satisfies

$$\mathbb{E}_{c^i \sim \mathcal{D}} ||\widehat{f} - f^*(c^i)||_1 \leq \mathbb{E}_{c^i \sim \mathcal{D}} ||\widetilde{f} - f^*(c^i)||_1 + O(1).$$

3. Faster Flows via Shorter Projection Paths

Algorithm 1 has made specific choices about finding projection paths, such as trying to send flows from excess to deficit nodes first, instead of to s, i.e., the algorithm prefers to amend the flow without reducing the actual flow value. In this section, we show such projection path choices do matter.

[3] Such assumptions are standard for the PAC-learning results in learning-augmented based run-time improvements, even if not explicitly stated. See, for instance (Sakaue and Oki, 2022).

We give an example particular graph structure, where, by careful selection of projection paths, one can restore feasibility of the given flow using significantly shorter projection paths, thus getting a more refined run time bound than that in Theorem 1.

Let $G = (V, E)$ be a directed graph, with $s, t \in V$. Suppose $V \setminus \{s, t\}$ forms a two-dimensional grid. Further for $u, v \in V \setminus \{s, t\}$, if $e = (u, v) \in E$ then the reverse direction edge $\overleftarrow{e} = (v, u) \in E$. We consider a pair of networks G^1, G^2 on G. The only difference in these networks is their capacity vectors, though we assume they have capacity vectors $c^1, c^2 \in \{1, M\}^m$ for some large M, and we assume that all edges incident to s or t have capacity M.

For $\ell \in [2]$, let the **boundary** $E_\ell = \{e \in E \mid c^\ell_e = 1\}$. We call G^ℓ **separable** if the vertices in $V \setminus \{s, t\}$ can be partitioned into subsets V_ℓ and $W_\ell = V \setminus (\{s, t\} \cup V_\ell)$, such that there is some $x \in V_1 \cap V_2$ with $(x, t) \in E$ and $y \in W_1 \cap W_2$ with $(s, y) \in E$, and for all $e = (u, v) \in E_\ell$, e has one endpoint in V_ℓ and the other in W_ℓ. Consider the set of vertices $\Delta = (V_2 \setminus V_1) \cup (W_2 \setminus W_1)$. This is the set of nodes that used to belong to V_1/W_1 in G_1, and yet has changed to be in W_2/V_2 in G_2 respectively, in other words, vertices that crossed the old boundary E_1. We say the transition between G^1 and G^2 is **d-local**, if for all pairs of distinct vertices u, v in Δ and its neighboring grids such that u, v are both in V_2/W_2, there exists a path from u to v composed of capacity M edges only. Later d will be used to control the length of the projection paths.

While we require additional assumptions for our theoretical results, these assumptions are sufficient but not necessary to take advantage of short projection paths. The image segmentation network in Section 4 is a generalization of this graph class. Our empirical results are consistent with the theory provided in this section.

Theorem 10 (Restates Theorem 2)**.** *Fix separable networks G^1 and G^2, where the transition between them is d-local. Any $\widehat{f}$ that is an optimal flow on G^1 can be used to warm-start Ford-Fulkerson on G^2 within time $O(d^2 \cdot |E| + d \cdot \eta(\widehat{f}))$.*

For the proof of Theorem 10, we first reroute flow along excess-deficit paths that does not cross the boundary E_2, until we only have nodes with excess on one side of the new boundary E_2 and deficits on the other. We argue that the rerouting uses paths of length $O(d)$. We then fix the excess and deficit nodes depending on whether they are in V_2 or W_2, by either sending flow from excess to deficit nodes directly, or use s, t to make up for excess/deficits. These paths could be longer, i.e. of length $O(|E|)$. We can argue the change in flow value is at most $||E_2| - |E_1|| \leq O(d^2)$, where the upper bound comes from he definition of d-local. This means the resulting flow, after feasibility projection, is only slightly sub-optimal.

Proof. For the network G^2 with capacity constraints c^2, project $\widehat{f}$ to satisfy c^2 as in Algorithm 1, and let the resulting flow be f. Note that excess/deficit can only arise because we have flow greater than 1 on an edge not in E_1 but is now in E_2. Further, they appear in pairs across the boundary with the same value.

To restore flow conservation on f, we can choose paths to first route flow from nodes with excess to deficit where both nodes are in V_2, and likewise in W_2. By our assumption that the networks are $d-$local, excess and deficit within V_2 or in W_2 have distance at most d. Since the capacity of the non-boundary edges is M, we can route flow until there is only excess or only deficit contained within V_2 and within W_2 with these projection paths.

Also, notice that, we can choose paths to perform re-routing so excess and deficit are symmetric across the boundary. Specifically, since excess/deficits appear in pairs before any re-routing, every time we route any flow to excess to deficit without crossing the boundary, go to the neighbors of these two nodes across the boundary and re-route the flow there to keep the symmetry. From the proof of Theorem 1 (see Appendix 6.1), we have $\mathsf{ex}_f + \mathsf{def}_f \leq \eta(\widehat{f})$, so the run-time of re-routing this excess and deficit is at most $O(d \cdot \eta(\widehat{f}))$.

Then, if V_2 contains a node with positive deficit and W_2 contains a node with positive excess and if there are any edges with positive flow going from V_2 to W_2, one can send flow on the reverse edge and remove that excess/ deficit with a projection path of length at most $2d$.

We will show any remaining deficit/ excess can be handled by paths using s and t. First, the max flow on G^1 has value $|E_1|$, since the edges crossing from W_1 into V_1 form a cut. On the other hand, for large M and by the existence of $x \in V_1 \cap V_2$ and $y \in W_1 \cap W_2$, there exists feasible flows with every edge (u, v) with $u \in W_1$ and $v \in V_1$ having flow 1 incoming to V_1. Second, our re-routing procedure made the excess within V_2 symmetric across the boundary to the deficit in W_2, so there is either positive excess in V_2 and positive deficit in W_2, or vice versa.

We use these two observations. Suppose that after re-routing, there is deficit inside of V_2 and excess outside of it. As a consequence of our routing, we can assume all edges with positive flow in E_2 are directed from W_2 to V_2. For every node u with positive excess in W_2, take the excess of u and send it to s, which is possible from the conditions for being separable. Similarly, for every node u with positive deficit inside of V_2, find paths from t to u and send $\mathsf{def}_f(u)$ from t to u, and this is again possible by the separable condition. The resulting flow is feasible. Further, there are no $s - t$ paths since all edges in E_2 going into V_2 are saturated and form a cut, so the flow is optimal. So re-routing this flow using s and t takes time $O(|E|(|E_1| - |E_2|))$.

When after re-routing there is excess inside of V_2 and deficit inside of W_2, the proof is similar. The edges with no excess/ deficit incident to them have flow 1 going into V_2. Since the deficit is outside of V_2, the boundary edges crossing from V_2 into W_2 have flow 1 going out of V_2. Send flow from s to the nodes with positive deficit inside of W_2, and send the excess inside of V_2 to t, which is possible by the conditions for being separable. The resulting flow is feasible, though perhaps not optimal, as one may need to saturate new boundary edges. The run-time is still $O(|E|(|E_2| - |E_1|))$.

By the definition of $d-$local, $||E_2| - |E_1|| \leq O(d^2)$. Therefore, the run-time of Algorithm 1 on these locally-changed instances is at most $O(d^2 \cdot |E| + d \cdot \eta(\widehat{f}))$. □

4. Empirical Results

In this section, we validate the theoretical results in Sections 2 and 3. We consider image segmentation, a core problem in computer vision that aims at separating an object from the background in a given image. The problem is re-formulated as a max-flow/ min-cut optimization problem in a line of work (Boykov and Jolly, 2001; Boykov and Kolmogorov, 2004; Boykov and Funka-Lea, 2006) and solved with combinatorial graph-cut algorithms, including Ford-Fulkerson.

We do not attempt to provide state-of-the-art run-time results on image segmentation. Our goal is to show that on real-world networks, warm-starting Ford-Fulkerson leads to big run-time improvements compared to cold-start Ford-Fulkerson. We highlight the following:

- For both Edmonds-Karp and Dinic's implementation of Ford-Fulkerson, warm-start offers improved running time compared to starting the algorithm from scratch (referred to as a **cold-start**).
- As we increase the number of image pixels (i.e., its resolution), the size of the constructed graph increases and the savings in time becomes more significant.
- The feasibility projection step in Algorithm 1 has high performance. It returns a feasible flow that is only slightly sub-optimal, and it finds short paths to fix the excess/deficits in doing so. Both factors contribute to warm-start being way more efficient than cold-start.

Datasets and data pre-processing We use four different image groups from the *Pattern Recognition and Image Processing* dataset from the University of Freiburg[4], named BIRDHOUSE, HEAD, SHOE and DOG respectively. The first three groups are from the dataset *Image Sequences*[5], in the format of .jpg images, whereas DOG, from *Stereo Ego-Motion Dataset*[6], is a video which we converted to .jpg.

[4]https://lmb.informatik.uni-freiburg.de/resources/datasets/

[5]https://lmb.informatik.uni-freiburg.de/resources/datasets/sequences.en.html

[6]https://lmb.informatik.uni-freiburg.de/resources/datasets/StereoEgomotion.en.html

(a) Birdhouse

(b) Head

(c) Shoe

(d) Dog

Figure 1. Examples of original images in each group.

Each image group contains a sequence of photos featuring the same object and background. The sequence may feature the object's motion relative to the background or changes in the camera's shooting angle. Any image is only slightly different from the previous one in the sequence, and this could potentially lead to minor differences in segmentation solutions. This justifies warm-starting with the optimal flow for the max flow problem found on the previous image.

Using data from a previous instance to warm-start an iterative process is one of the most common ways to actually warm-start algorithms practically. We see our empirical choice of prediction as simple, yet effective, and also indicative of what practitioners do. Moreover, other theoretically-focused works have chosen such predictions (see for instance the paper by Chen et al. (2022a) who use prior optimal dual values from past data to warm-start their primal-dual method on current data). The experiments give empirical evidence that learning flows is reasonable, while the PAC-learning results give the theoretical evidence, thus complementing each other.

Table 1. Image groups desciption

Image Group	Object, background	Original size	Cropped size
BIRDHOUSE	wood birdhouse, backyard	1280, 720	600, 600
HEAD	a person's head, buildings	1280, 720	600, 600
SHOE	a shoe, floor and other toys	1280, 720	600, 600
DOG	Bernese Mountain dog, lawn	1920, 1080	500, 500

We take 10 images from each group, cropped them to be 600×600 pixels with the object included, and gray-scaled them. Then we resize the images to generate image sequences of different sizes. See Table 1 for detailed information about the image groups, the featured object/backround, and the original and cropped sizes of each image. See Figures 1 and 2 for an image instance from each group.

Graph construction Following the practice in Boykov and Funka-Lea (2006), we briefly describe how to formulate image segmentation as a max-flow/min-cut problem and

(a) BH (b) Head (c) Shoe (d) Dog

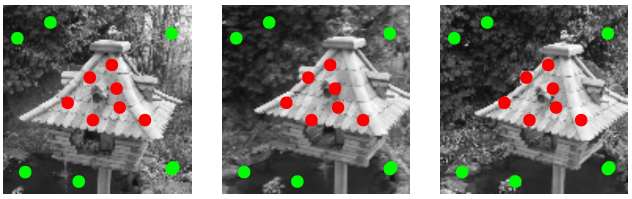
Figure 2. Cropped, gray-scaled images in each group.

how to write the boundary-based objective function. Our input is an image with pixel set V, along with two sets of **seeds** $\mathcal{O}, \mathcal{B}$, which are pixels predetermined to be part of the object or background, respectively (often selected by human experts), to make the segmentation task easier. Let I_v denote the **intensity** (or gray scale) of pixel v. For any two pixels p, q, separating them in the object/background segmentation solution induces a **penalty** of $\beta_{p,q}$. If p, q are neighboring pixels, i.e. p and q are either in the same column and in adjacent rows or same row and adjacent columns, then $\beta_{p,q} = C \exp(-\frac{(I_p - I_q)^2}{2\sigma^2})$, where C is a relatively big constant scalar, otherwise it is 0. Thus $\beta_{p,q}$ gets bigger when there is low contrast between neighboring p and q.

For a given solution let J denote the object pixels. The **boundary-based** objective function is the summation of the penalties over all pairs of pixels: $\max_J \sum_{p \in J, q \notin J} \beta_{p,q}$, for J satisfying $\mathcal{O} \subseteq J, \mathcal{B} \subseteq V \setminus J$. Penalties are only imposed on the object boundary. The best segmentation minimizes the total penalty, thus maximizing the contrast between the object and background across the boundary, while satisfying the constraints imposed by seeds.

This is equivalent to solving the max-flow/min-cut problem on the following graph. Let the node set be all the pixels plus two terminal nodes: the object terminal s (source) and the background terminal t (sink). We add the following arcs: (1) from s to every node in $\mathcal{O}$, with a huge capacity M; (2) from every node in $\mathcal{B}$ to t, again with capacity M; (3) from every pair of node $p, q \in V$ (including the seeds), both arcs (p, q) and (q, p) with capacity $\beta_{p,q}$. The value M should ensure that these arcs never appear in the optimal cut. The flow goes from s to t. For an $n \times n$ pixels image, the graph is sparse with $O(n^2)$ nodes and also $O(n^2)$ arcs.

Link to theory For an image sequence, the constructed graphs are a generalization of the setting in Section 3. The graphs form 2-dimensional grids and share the same network structure, the only differences being the capacity vectors. In addition, Section 3 makes other assumptions which also translate into properties of the images. The 1 or M edge capacities assumption implies an extreme contrast between the gray scales of object and background pixels. The d-local assumption says that from one image to the next, the new object and background pixels are geographically close, implying only minor changes in the object's shape and location. Our image sequences do not strictly satisfy these properties. However, in all of our experiments the conclusions remain robust against moderate violations of the theoretical assumptions, showing that warm-starts can be beneficial in practice beyond current theoretical limits.



(a) Image 1 (b) Image 5 (c) Image 10

Figure 3. Seeds on the first, fifth and last images from 120×120 pixels BIRDHOUSE. Red for object, green for background.

(a) Image 1 (b) Image 5 (c) Image 10

Figure 4. Cuts (red) on the first, fifth and last images from 120×120 pixels BIRDHOUSE

Detailed experiment settings Each image sequence has 10 images and they share the same set of seeds, so the constructed graphs have the same structure. See Figure 3 for seeds for BIRDHOUSE. Starting with the second image, we reuse the old max-flow solution on the previous one and pass the flow to Algorithm 1. During the feasibility projection, we pick a node and keep diminishing its excess/deficit by finding a projection path and sending flow down that path, until excess/deficit is 0. As in Section 3, we prioritize projection paths excluding s and t, since these modifications preserve the overall flow value, and we only sending flow back to s and from t when no other paths exist.

We compare cold- and warm-start for both Edmonds-Karp and Dinic's algorithms. Recall that warm-start also needs to find paths to restore excess/deficit (referred to as projection paths). We use breadth-fist-search (BFS) to find such projection paths in our warm-starts for both Edmonds-Karp and Dinic's. We use the BFS procedure for our warm-started Dinic's instead of the expected subroutine from Dinic's algorithm because the overhead of building the level graph is more time consuming than running BFS. This is due to the projection paths being short.

We use $n \times n$ pixel images for $n \in \{30, 60, 120\}$. Numerically, the σ in the definition of $\beta_{\cdot,\cdot}$ is 50, and C is 100. To make the capacities integral, all $\beta_{p,q}$'s are rounded down to the nearest integer. Notice that $\beta_{p,q} \leq C$ by definition. We let $M = C|V|^2$ to make the term sufficiently large.

All experiments are run on a device with Intel(R) Core(TM)

i7-7600U CPU @ 2.80GHz, with 24G memory. We record the wall-clock running time for both algorithms. Many of the image process tools and functions are based on the *Image Segmentation* Github project (Jiang, 2017).

Table 2. Average running times of cold-/warm-start Ford Fulkerson and the percentage of time saved by warm-start, Edmonds-Karp

Image Group	30×30	60×60	120×120
BIRDHOUSE	0.83/0.55, 34.07%	8.48/3.48, 58.98%	109.06/37.31, 65.78%
HEAD	0.65/0.45 31.06%	9.52/4.28, 55.07%	112.66/31.77, 71.80%
SHOE	0.72/0.46, 36.01%	8.81/3.04, 65.47%	111.05/30.44, 72.59%
DOG	0.73/0.41, 42.96%	22.38/6.89, 69.22%	202.99 / 42.04, 79.29%

Table 3. Average running times of cold-/warm-start Ford Fulkerson and the percentage of time saved by warm-start, Dinic

Image Group	30×30	60×60	120×120
BIRDHOUSE	0.38/0.37, 2.49%	5.81/3.17, 45.43%	82.52/35.37, 57.14%
HEAD	0.36/0.36 0.58%	7.7/4.44, 42.35%	149.12/49.44, 62.88%
SHOE	0.39/0.37, 5.07%	7.01/3.35, 52.24%	140.52/49.33, 64.9%
DOG	0.5/0.41, 10.16%	12.38/4.99, 59.66%	206.85 / 58.98, 71.48%

Results We first show that the boundary-based image segmentation approach generates reasonable cuts. For example, Figure 4 illustrates cuts from the 120×120 BIRDHOUSE sequence. See Appendix 6.3 for other examples. We then compare the running time of cold- and warm-start Ford-Fulkerson. As all algorithms are returning optimal flows, there are no qualitative aspects of the solutions to measure. Table 2 and 3 show results in all experiments settings for Edmonds-Karp and Dinic, rows being image groups and columns image sizes. Each entry is formatted as "cold-start time (s) / warm-start time(s), warm-start time savings (%)".

These results show warm-starting Ford-Fulkerson greatly improves the efficiency in all settings. Further, both cold- and warm- start's running time increases polynomially with the image width n, but warm-start grows slower, making it a potentially desirable approach on large scale networks. This is most obvious on image group DOG using Edmonds-Karp, where warm-start time is 60% of cold-start time on 30×30 pixels versus 20% on 120×120 pixels. These conclusions hold for both Edmonds-Karp and Dinic, with Dinic being slightly more efficient on smaller datasets.

Next we examine the execution of cold-/warm-start in more detail, taking the 120×120 BIRDHOUSE sequence for Edmonds-Karp for example (Table 4). The table gives the average length of the augmenting paths ('avg length') and the average number of paths found ('avg #') over the sequence of images. See Appendix 6.3 for complete data.

Table 4. Comparison of projection and augmenting paths in cold- and warm-start Ford-Fulkerson, the first 5 images from the 120×120 BIRDHOUSE image sequence

Image #	cold-start aug path #	cold-start aug path avg length	warm-start proj path #	warm-start proj path avg length	warm-start aug path #	warm-start aug path avg length
1	2453	67.93	2105	9.39	628	81.48
2	2093	65.22	3393	19.28	0	0
3	2536	74.88	2038	9.71	896	101.731
4	2089	69.09	3335	28.55	0	0
5	1908	68.53	3226	22.97	0	0

Results in Table 4 suggest that the projected feasible flow is in general only slightly sub-optimal, which is key for warm-starts efficiency. Max-flow on the previous image is a good starting point for warm-start with the feasibility projection algorithm. On average, after rounding down the previous max-flow to satisfy the new edge capacities, the total excess/deficit is (1.75 ± 0.44) % of the real maximum flow value. Moreover, fixing the excess/deficit results in a near optimal flow. Indeed, the projection quickly gives a feasible flow that recovers $(96 \pm 6)\%$ of the maximum flow.

Another factor contributing to the efficiency of warm-start is the projection path-finding subroutine. Recall that both cold- and warm-start use the same BFS subroutine to find either an s, t augmenting path or a projection path. The theory in Section 3 suggests that paths in the projection step will take less time to find. To show this empirically, we collected data on the number of augmenting/feasibility projection paths found and their average lengths for both cold- and warm-start. Overall, compared with cold-start, warm-start has shorter projection paths on average, suggesting massive savings in the BFS running time per path. While we show this for the BIRDHOUSE images in Table 4, this is true on other datasets too, available in Appendix 6.3. This explains the efficiency even if the excess/deficit is large. This shows that the theoretical expectations raised in Section 3 are predictive of empirical performance.

5. Conclusion

We show how to warm-start the Ford-Fulkerson algorithm for computing flows, as well as prove strong theoretical results and give empirical evidence of good performance of our algorithm. We further refine our analysis to capture the gains due to using *short* projection paths to route excess flow and show that these scenarios are prevalent in image segmentation applications. Many interesting challenges remain. For one, there are many known algorithms for computing flows, and it would be interesting to see if those methods can also be sped up in a similar fashion. A technical roadblock lies in handling both under- and over- predictions, particularly when predictions lead to infeasible flows. More generally, a network flow problem can be written as a linear program. Another direction is finding algorithms for solving general LPs that can be helped by judiciously chosen predictions.

Acknowledgements

Sami Davies was supported by an NSF Computing Innovation Fellowship. Benjamin Moseley was supported in part by a Google Research Award, an Informs Research Award, a Carnegie Bosch Junior Faculty Chair, and NSF grants CCF-2121744 and CCF-1845146.

References

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice hall.

Altner, D. S. and Ergun, Ö. (2008). Rapidly solving an online sequence of maximum flow problems with extensions to computing robust minimum cuts. In Perron, L. and Trick, M. A., editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 5th International Conference, CPAIOR 2008, Paris, France, May 20-23, 2008, Proceedings*, volume 5015 of *Lecture Notes in Computer Science*, pages 283–287. Springer.

Antoniadis, A., Gouleakis, T., Kleer, P., and Kolev, P. (2020). Secretary and online matching problems with machine learned advice. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Balcan, M., DeBlasio, D. F., Dick, T., Kingsford, C., Sandholm, T., and Vitercik, E. (2021). How much data is sufficient to learn high-performing algorithms? generalization guarantees for data-driven algorithm design. In Khuller, S. and Williams, V. V., editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 919–932. ACM.

Bhadra, S., Kundu, A., and Khatua, S. (2020). Optimization of road traffic congestion in urban traffic network using dinic's algorithm. In Abraham, A., Sasaki, H., Rios, R., Gandhi, N., Singh, U., and Ma, K., editors, *Innovations in Bio-Inspired Computing and Applications - Proceedings of the 11th International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2020) held during December 16-18, 2020*, volume 1372 of *Advances in Intelligent Systems and Computing*, pages 372–379. Springer.

Boykov, Y. and Funka-Lea, G. (2006). Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131.

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137.

Boykov, Y. Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE.

Chandran, B. G. and Hochbaum, D. S. (2009). A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. *Operations research*, 57(2):358–376.

Chen, J. Y., Silwal, S., Vakilian, A., and Zhang, F. (2022a). Faster fundamental graph algorithms via learned predictions. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S., editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 3583–3602. PMLR.

Chen, L., Kyng, R., Liu, Y. P., Peng, R., Gutenberg, M. P., and Sachdeva, S. (2022b). Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623.

Cherkassky, B. V. and Goldberg, A. V. (1997). On implementing the push—relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410.

Dinitz, M., Im, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. (2021). Faster matchings via learned duals. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 10393–10406.

Dinitz, Y. (2006). Dinitz'algorithm: The original version and even's version. *Theoretical Computer Science: Essays in Memory of Shimon Even*, pages 218–240.

Edmonds, J. and Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264.

Ford, L. R. and Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404.

Gupta, R. and Roughgarden, T. (2017). A PAC approach to application-specific algorithm selection. *SIAM J. Comput.*, 46(3):992–1017.

Im, S., Kumar, R., Petety, A., and Purohit, M. (2022). Parsimonious learning-augmented caching. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S., editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9588–9601. PMLR.

Im, S., Kumar, R., Qaem, M. M., and Purohit, M. (2021a). Non-clairvoyant scheduling with predictions. In Agrawal, K. and Azar, Y., editors, *SPAA '21: 33rd ACM Symposium*

on Parallelism in Algorithms and Architectures, Virtual Event, USA, 6-8 July, 2021, pages 285–294. ACM.

Im, S., Moseley, B., and Pruhs, K. (2021b). The matroid intersection cover problem. *Oper. Res. Lett.*, 49(1):17–22.

Jiang, J. (2017). Image segmentation. https://github.com/julie-jiang/image-segmentation/.

Kelner, J. A., Lee, Y. T., Orecchia, L., and Sidford, A. (2014). An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In Chekuri, C., editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 217–226. SIAM.

Kraska, T., Beutel, A., Chi, E. H., Dean, J., and Polyzotis, N. (2018). The case for learned index structures. In Das, G., Jermaine, C. M., and Bernstein, P. A., editors, *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 489–504. ACM.

Lattanzi, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. (2020). Online scheduling via learned weights. In Chawla, S., editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1859–1877. SIAM.

Lattanzi, S., Moseley, B., Vassilvitskii, S., Wang, Y., and Zhou, R. (2021). Robust online correlation clustering. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 4688–4698.

Lavastida, T., Moseley, B., Ravi, R., and Xu, C. (2021). Learnable and instance-robust predictions for online matching, flows and load balancing. In Mutzel, P., Pagh, R., and Herman, G., editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 59:1–59:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Lindermayr, A. and Megow, N. (2022). Permutation predictions for non-clairvoyant scheduling. In Agrawal, K. and Lee, I. A., editors, *SPAA '22: 34th ACM Symposium on Parallelism in Algorithms and Architectures, Philadelphia, PA, USA, July 11 - 14, 2022*, pages 357–368. ACM.

Lu, P., Ren, X., Sun, E., and Zhang, Y. (2021). Generalized sorting with predictions. In Le, H. V. and King, V., editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 111–117. SIAM.

Lykouris, T. and Vassilvitskii, S. (2021). Competitive caching with machine learned advice. *J. ACM*, 68(4):24:1–24:25.

Mitzenmacher, M. and Vassilvitskii, S. (2022). Algorithms with predictions. *Commun. ACM*, 65(7):33–35.

Polak, A. and Zub, M. (2022). Learning-augmented maximum flow. *CoRR*, abs/2207.12911.

Purohit, M., Svitkina, Z., and Kumar, R. (2018). Improving online algorithms via ML predictions. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9684–9693.

Sakaue, S. and Oki, T. (2022). Discrete-convex-analysis-based framework for warm-starting algorithms with predictions. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022,*.

Sherman, J. (2013). Nearly maximum flows in nearly linear time. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 263–269. IEEE Computer Society.

Vineet, V. and Narayanan, P. (2008). Cuda cuts: Fast graph cuts on the gpu. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE.

6. Appendix

6.1. Omitted Proofs

Proof of [Lemma 4] Assume for sake of deriving a contradiction that (without loss of generality) sending flow from some $u \in A'_f$ to s formed flow h, for which there exists a path from A'_f to B'_f with positive capacity in G_h. Let $p_1 = (u_1, \ldots, s)$ be the path along which flow was sent in G_f, and let $p_2 = (u_2, \ldots, v)$ be the path with positive capacity in G_h. Note that $u_1, u_2 \in A'_f$ and $v \in B'_f$ by the assumptions. Since p_2 went from having 0 capacity in G_f to positive capacity in G_h, at least one edge $e = (a, b)$ of p_2 has $\overleftarrow{e} = (b, a)$ in p_1. If there are multiple such edges take e to be the last such edge in p_1. Let $p'_1 = (u_1, \ldots, a)$ be the truncation of p_1 at a and let $p'_2 = (a, \ldots, v)$ be the suffix of p_2 that begins at a. Then let p' be the concatenation of p'_1 and p'_2. Note that p'_1 and p'_2 both have positive capacity in G_f. Thus p' is a path in G_f with positive capacity from u to v, which is a contradiction. □

Proof of [Theorem 1]

Given a flow $\widehat{f}$ that does not satisfy capacity constraints, Algorithm 1 simply updates the edges $E' \subseteq E$ that violate capacity $\widehat{f}_e > c_e$ to $\widehat{f}_e \leftarrow c_e$ for $e \in E'$. This can be done in time $O(|E'|)$. Further, rounding down the flow on these edges changes the value of the flow and the sum of the excess and deficit by at most $\sum_e \max\{\widehat{f}_e - c_e, 0\}$.

In Section 2.1, we showed that given $\widehat{f}$ that satisfies capacity constraints, Algorithm 1 will find an optimal f^*. Next, we analyze the run-time of Algorithm 1. Each iteration of the main while loop in Algorithm 1 costs time $O(|E|)$. Further, the number of iterations in the main while loop in Algorithm 1 is at most $\mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}}$. Let f be the feasible flow obtained by Algorithm 1 at the end of the main while loop. The run-time to produce flow f is $O(|E|(\mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}}))$.

At most $|\nu(f^*) - \nu(f)|$ iterations of any Ford-Fulkerson procedure are needed to arrive at the optimal flow value $\nu(f^*)$ from f by Proposition 3. Each iteration of Ford-Fulkerson also costs $O(|E|)$.

Therefore, run-time of Algorithm 1 given a prediction which satisfies capacity constraints is at most

$$O(|E| \cdot (|\nu(f) - \nu(f^*)| + \mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}})).$$

Combining this with the loss from projecting to satisfy capacity constraints, the full run-time of Algorithm 1 is

$$O\big(|E| \cdot \big(\sum_e \max\{\widehat{f}_e - c_e, 0\} + |\nu(f) - \nu(f^*)| + \mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}}\big)\big).$$

We will show all of the terms multiplying $|E|$ in the above can be bounded by $O(\eta(\widehat{f}))$ (recall this was the statement of Lemma 7).

Proof of Lemma 7. First, we see that $|\nu(f) - \nu(f^*)|$ and $|\nu(\widehat{f}) - \nu(f^*)|$ can only differ by however much value was lost and however much excess and deficit was gained in projecting $\widehat{f}$ to the feasible flow f. Therefore, we can upper bound $|\nu(f) - \nu(f^*)|$ by

$$|\nu(f) - \nu(f^*)| \leq |\nu(\widehat{f}) - \nu(f^*)| + \sum_e \max\{\widehat{f}_e - c_e, 0\} + \mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}}.$$

Then, we can further upper bound $|\nu(\widehat{f}) - \nu(f^*)|$ by rewriting the difference between the values of the flows

$$|\nu(f^*) - \nu(\widehat{f})| = \min_{f^* \in \mathcal{F}^*} \left| \sum_{e=(s,v)} f^*_e - \sum_{e=(s,v)} \widehat{f}_e \right| \leq \eta(\widehat{f}).$$

The next term to bound in terms of the ℓ_1 error is $\sum_e \max\{\widehat{f}_e - c_e, 0\}$, though it is straight forward to see

$$\sum_e \max\{\widehat{f}_e - c_e, 0\} \leq \min_{f^* \in \mathcal{F}^*} \sum_e \max\{\widehat{f}_e - f^*_e, 0\} \leq \eta(\widehat{f}).$$

Lastly, we see that the excess/ deficit of any node $v \in V \setminus \{s, t\}$ can be charged to the difference between $\widehat{f}_e$ and f^*_e for e adjacent to v, as any $f^* \in \mathcal{F}^*$ has excess/ deficit 0 on all non-source and sink nodes. Therefore, $\mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}} \leq \eta(\widehat{f})$. □

The justification that the worst-case run-time of Algorithm 1 is $O(T)$, for $O(T)$ the worst-case run-time of the chosen Ford-Fulkerson implementation, is in Section 2.

□

6.2. PAC-Learnability

Let $c^1, \ldots, c^s$ from $\mathcal{D}$ be the samples, and $\mathcal{F}$ be the set of integral flows on G satisfying the capacities in c. In what follows, let $f^*(c^i)$ be the uniquely associated optimal flow on the sample $c^i \sim \mathcal{D}$ (recall Assumption 8).

Let $\widehat{f}$ denote a predicted flow. When our goal is to warm-start Ford-Fulkerson, we choose the predicted flow to be that in $\mathcal{F}$ which minimizes the empirical risk $\widehat{f} = \text{argmin}_{f \in \mathcal{F}} \frac{1}{s} \sum_{j=1}^{s} ||\widehat{f} - f^*(c^j)||_1$. That choice corresponds to the flow minimizing the run-time given s samples, as shown in Lemma 11.

Lemma 11. *One can find a flow $\widehat{f} \in \mathcal{F}$ minimizing $\frac{1}{s} \sum_{j=1}^{s} ||\widehat{f} - f^*(c^j)||_1$ from independent samples $c^1, \ldots, c^s \sim \mathcal{D}$ in polynomial time by taking $\widehat{f}_e = \textsf{median}(f^*(c^1)_e, \ldots, f^*(c^s)_e)$ for all $e \in E$.*

Proof. We would like to find $\widehat{f} \in \mathcal{F}$ that minimizes $\frac{1}{s} \sum_{j=1}^{s} ||\widehat{f} - f^*(c^j)||_1$. Since we do not require flow conservation, the minimization can occur over each edge independently, where $\widehat{f}_e$ will be in $[0, c_e]$, i.e. it suffices to minimize $\frac{1}{s} \sum_{j=1}^{s} |\widehat{f}_e - f^*(c^j)_e|$ for each $e \in E$. The function $\frac{1}{s} \sum_{j=1}^{s} |\widehat{f}_e - f^*(c^j)_e|$ is continuous and piece-wise linear in $\widehat{f}_e$, where the slope changes at the points $\{f^*(c^j)_e\}_j$. It is well-known that the minimum of this function in $[0, c_e]$ is $\textsf{median}(f^*(c^1)_e, \ldots, f^*(c^s)_e)$. □

In the proof of Theorem 9, we will use some well-known results regarding the pseudo-dimension of a class of functions.

The VC dimension is a quantity that captures the complexity of a family of binary functions, and the pseudo-dimension is the analog of this for real-valued functions Specifically, the *pseudo-dimension* of a family of real-valued functions $\mathcal{H}$ is the largest sized subset shattered by $\mathcal{H}$. A subset $S = \{x_1, \ldots, x_s\}$ of X is *shattered* by $\mathcal{H}$ if there exists real-valued witnesses $r_1, \cdots, r_s$ such that for each of the 2^s subsets T of S, there exists a function $h \in \mathcal{H}$ with $h(x_i) \leq r_i$ if and only if $i \in T$.

The following theorem relates the convergence of the sample mean of some $h \in \mathcal{H}$ to its expectation, and this relation depends on the pseudo-dimension.

Theorem 12 (Uniform convergence)**.** *Let $\mathcal{H}$ be a class of functions with domain X and range in $[0, H]$. Let $d_{\mathcal{H}}$ be the pseudo-dimension of $\mathcal{H}$. For every distribution $\mathcal{D}$ over X, every $\epsilon > 0$, and every $\delta \in (0, 1]$, if*

$$s \geq c(H/\epsilon)^2(d_{\mathcal{H}} + \ln(1/\delta))$$

for some constant c, then with prob at least $1 - \delta$ over s samples $x_1 \ldots, x_s \in \mathcal{D}$,

$$\left| \left(\frac{1}{s} \sum_{i=1}^{s} h(x_i) \right) - \mathbb{E}_{x \sim \mathcal{D}}[h(x)] \right| < \epsilon.$$

Equipped with Theorem 12, we are ready to prove our PAC-learning result. We note that this proof structure exactly follows that in (Dinitz et al., 2021).

Theorem 13. *[Restates Theorem 9] Let $c^1, \ldots, c^s$ be sampled i.i.d. from $\mathcal{D}$ and let $\widehat{f} = \text{argmin}_{f \in \mathcal{F}_0} \frac{1}{s} \sum_{j=1}^{s} ||f - f^*(c^j)||_1$. For*

$$s = \Omega((\max_e c_e^2 \cdot m^2)(m \log m + \log(1/p)))$$

and $\widetilde{f} = \text{argmin}_{f \in \mathcal{F}_0} \mathbb{E}_{c^i \sim \mathcal{D}} ||f - f^*(c^i)||_1$, then with probability at least $1 - p$, $\widehat{f}$ satisfies

$$\mathbb{E}_{c^i \sim \mathcal{D}} ||\widehat{f} - f^*(c^i)||_1 \leq \mathbb{E}_{c^i \sim \mathcal{D}} ||\widetilde{f} - f^*(c^i)||_1 + O(1).$$

Proof. We will construct a class of functions that contains the loss functions of the flow f given capacity constraints c^i. Then, we will apply Theorem 12 to this class of functions.

For every integral flow $f \in \mathbb{R}^{|E|}$ that satisfies the capacity vector c^i, let the function $g_f(c^i) = ||f^*(c^i) - f||_1$ be the loss function of f on c^i. Then let $\mathcal{H} = \{g_f \mid f \in \mathbb{R}^m\}$ be the family of all of these loss functions.

We saw in Lemma 11 how to efficiently compute the empirical risk minimizer. Also, the upper bound of the range of the loss functions, i.e. H in the statement of Theorem 12, is at most $m \cdot \max_e c_e$. To prove our lemma, it remains to bound the pseudo-dimension of $\mathcal{H}$.

We will upper bound the pseudo-dimension of $\mathcal{H}$ by showing it is no more than the pseudo-dimension of another class of functions, $\mathcal{H}_m$, whose pseudo-dimension is already known. Let $\mathcal{H}_m = \{h_y \mid y \in \mathbb{R}^m\}$ for $h_y(x) = ||y - x||_1$. The following result appears as Theorem 19 in (Dinitz et al., 2021), and the reader may refer to that paper for its proof.

Lemma 14. *The pseudo-dimension of $\mathcal{H}_m$ is at most $O(m \log m)$.*

Now all that remains is to prove the following lemma, relating the pseudo-dimensions of the two classes.

Lemma 15. *If the pseudo-dimension of $\mathcal{H}_m$ is at most d, then the pseudo-dimension of $\mathcal{H}$ is at most d.*

Proof. Let $S = \{c^1, \ldots, c^d\}$ be a set that is shattered by $\mathcal{H}$. Let $r_1, \ldots, r_d \in \mathbb{R}$ be the witnesses such that for all $S' \subseteq [d]$, there exists some $g_{f^{S'}} \in \mathcal{H}$ with $g_{f^{S'}}(c^j) = ||f^*(c^j) - f^{S'}||_1 \leq r_j$ if and only if $j \in S'$.

We will construct a set $\widetilde{S}$ of size d from S that is shattered by $\mathcal{H}_m$. Let $\widetilde{S} = \{f^*(c^1), \ldots, f^*(c^d)\}$ and fix some $S' \subseteq [d]$. Then $h_{f^{S'}}(f^*(c^j)) = ||f^{S'} - f^*(c^j)||_1 \leq r_j$ if and only if $j \in S'$.

□

Plugging $H \leq m \cdot \max_e c_e$ and $d_{\mathcal{H}} \leq O(m \log m)$ into Theorem 12, we see that it suffices to take

$$s \geq \Omega((\max_e c_e \cdot m/\epsilon)^2 (m \log m + \ln(1/\delta))).$$

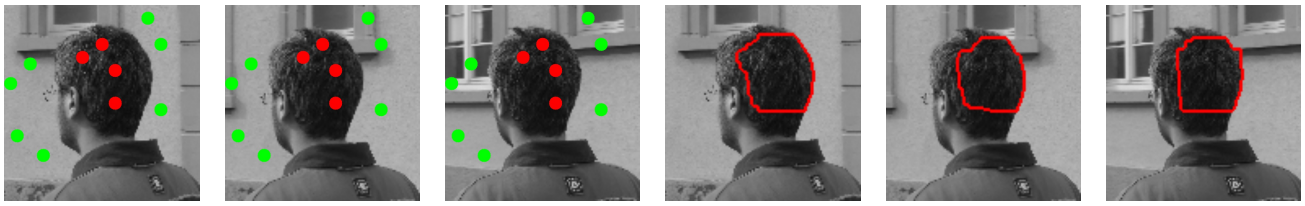
□

6.3. More Experimental Results

In this section, we give a more detailed description of the experiment settings and provide more complete collected data and results.

More on choice of seeds and cuts Recall that on the 10 images from the same sequence, the **seed** pixels are always fixed. We note here the choice of seeds (number of seeds and their locations) affects which min-cut solution is found a lot. However, as long as the seeds give a reasonable solution that is close to the real object/background boundary, the conclusions in the comparison between cold- and warm-start remain robust against a change of seeds.

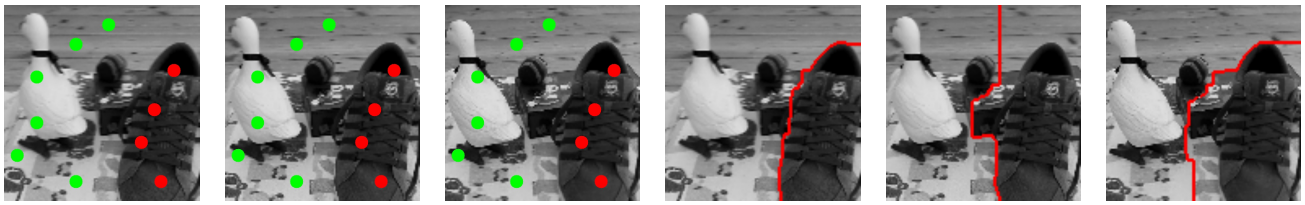
In Section 4, we showed seeds and optimal cuts on images 1, 5, and 10 of the 120×120 pixel BIRDHOUSE sequence in Figures 3 and 4. Here we show, in addition, our seeds and resulting cuts on images 1, 5, and 10 of the other 120×120 sequences. Those of HEAD are in Figure 5, those of SHOE in Figure 6, and those of DOG in Figure 7.



(a) Image 1, seeds (b) Image 5, seeds (c) Image 10, seeds (d) Image 1, cut (e) Image 5, cut (f) Image 10, cut

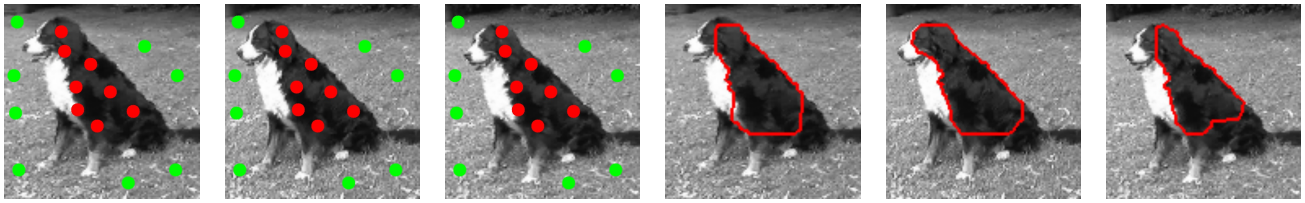
Figure 5. Seeds and resulting cuts on the first, fifth and last images from 120×120 pixels HEAD. Red seeds for object, green seeds for background, red line for cut.

When we select a seed, we draw a two-dimensional ball around the target seed and let every pixel in this ball be a seed as well. We found this practice to work better than simply choosing individual pixels as seeds. When we switch from



(a) Image 1, seeds (b) Image 5, seeds (c) Image 10, seeds (d) Image 1, cut (e) Image 5, cut (f) Image 10, cut

Figure 6. Seeds and resulting cuts on the first, fifth and last images from 120×120 pixels SHOE. Red seeds for object, green seeds for background, red line for cut.



(a) Image 1, seeds (b) Image 5, seeds (c) Image 10, seeds (d) Image 1, cut (e) Image 5, cut (f) Image 10, cut

Figure 7. Seeds and resulting cuts on the first, fifth and last images from 120×120 pixels DOG. Red seeds for object, green seeds for background, red line for cut.

low-resolution (30×30) to high-resolution (120×120) images, we rescale the radius of this ball proportional to the number of pixels on each side. On the 30×30, 60×60, 120×120 pixel images, the ball's radius is 1, 2 and 4 pixels, respectively. In other words, if we stretch/compress the images of different resolution to be the same size, the ball will roughly have the same area geometrically. We also found this to be more effective than fixing the pixel radius, despite the change in resolution.

(a) 30×30 (b) 60×60 (c) 120×120

Figure 8. Area that each seed covers on the same image with different resolutions.

Note that although the location of the seeds remains unchanged throughout an image sequence, we may still need to provide more seeds when we switch from low- to high-resolution images. Intuitively, blurring the image lessens the minor contrast of pixels within the object and makes the geometric shape easier to capture.

The seeds and min-cut results on the 30×30 and 60×60 sequences can be found in the code directory uploaded in the supplementary folder.

More on the warm-start magic In the main body we gave evidence—both theoretically and empirically—that the savings in the run-time of warm-start is mostly due to:

- The algorithm's ability to use short projection paths to re-route excess flow to nodes with deficit flow, thus projecting the predicted flow to a feasible one quickly.

- An only slightly sub-optimal flow after the feasibility projection, so that warm-start takes fewer augmenting paths to reach an optimal flow.

Here we provide more results in support of these two claims.

To show the level of total excess/deficit (whichever one is larger) and the flow value after the feasibility projection step, we show two ratios: total excess/deificit over max-flow (Table 5), and feasible flow value over max-flow (Table 6). One can see that typically the total excess/deficit is not negligible. In fact they are quite high and if the algorithm does not resolve excesses/deficits in the right way (such as sending all excess to the source) it could cause the flow value to diminish a lot. Our feasibility projection makes good decisions about using projection paths to make up for excess/deficit, so that it outputs a feasible flow with almost optimal flow value.

Table 5. Average ratio of total excess/deficit over max-flow value in warm-start

Image Group	30×30	60×60	90×90
BIRDHOUSE	1.06 ± 0.22	1.60 ± 0.21	1.75 ± 0.44
HEAD	0.49 ± 0.12	0.6 ± 0.12	0.74 ± 0.1
SHOE	0.49 ± 0.13	0.66 ± 0.08	0.95 ± 0.14
DOG	0.55 ± 0.07	0.8 ± 0.07	1.08 ± 0.19

Table 6. Average ratio of flow value after feasibility projection over max-flow value in warm-start

Image Group	30×30	60×60	90×90
BIRDHOUSE	0.94 ± 0.09	0.98 ± 0.03	0.96 ± 0.06
HEAD	0.98 ± 0.03	0.98 ± 0.03	0.99 ± 0.01
SHOE	0.98 ± 0.02	0.98 ± 0.03	0.98 ± 0.02
DOG	0.97 ± 0.04	0.97 ± 0.03	0.98 ± 0.03

To show that the conclusion of projection paths being short broadly holds for all image groups, we give the average length of the augmenting and projection paths ('avg length') and the number of paths found ('aug path #' and 'proj path #') over the first 5 images in the sequence for the 120×120 HEAD sequence in Table 7, the 120×120 SHOE sequence in Table 8, and the 120×120 DOG sequence in Table 9. Note the analogous table for the 120×120 BIRDHOUSE sequence (Table 4) is in Section 4.

Table 7. Comparison of projection and augmenting paths in cold- and warm-start Ford-Fulkerson, the first 5 images from the 120×120 HEAD image sequence

Image #	cold-start aug path #	cold-start aug path avg length	warm-start proj path #	warm-start proj path avg length	warm-start aug path #	warm-start aug path avg length
1	2714	82.65	2573	15.93	221	80.42
2	2687	82.74	2512	20.40	217	135.68
3	2475	76.63	2667	19.78	0	0
4	2379	76.44	2140	17.00	0	0
5	2349	75.66	2260	19.97	112	138.14

Further, we show the equivalence of these tables for the other two image sizes/resolutions, 30×30 and 60×60, for image groups HEAD (Table 10 and 11) and SHOE (Table 12 and 13). For these two groups, sequences of all three sizes share the same location of seeds. One can see that, the average length of augmenting path in cold-start Ford-Fulkerson grows roughly proportional to the width of the image. The average length of projection path during the warm-start feasibility projection also grows as the width of the image grows, but slightly slower than the former. This could potentially cause warm-start to be more advantageous on high-resolution images.

The omitted data tables and other experiment results can be found in the uploaded program directory (see the README.md file for instructions).

Table 8. Comparison of projection and augmenting paths in cold- and warm-start Ford-Fulkerson, the first 5 images from the 120×120 SHOE image sequence

Image #	cold-start aug path #	cold-start aug path avg length	warm-start proj path #	warm-start proj path avg length	warm-start aug path #	warm-start aug path avg length
1	1948	89.23	2252	22.70	0	0
2	2081	91.67	1992	16.54	112	148.41
3	2039	93.88	1936	14.91	177	142.51
4	2110	101.97	2525	35.04	0	0
5	2016	93.68	2375	18.60	0	0

Table 9. Comparison of projection and augmenting paths in cold- and warm-start Ford-Fulkerson, the first 5 images from the 120×120 DOG image sequence

Image #	cold-start aug path #	cold-start aug path avg length	warm-start proj path #	warm-start proj path avg length	warm-start aug path #	warm-start aug path avg length
1	3314	63.04	3684	12.51	0	0
2	3200	65.56	4611	21.69	0	0
3	3138	63.53	3515	12.30	0	0
4	3259	66.61	3270	10.74	444	87.08
5	3120	64.43	3932	12.63	0	0

Table 10. Comparison of projection and augmenting paths in cold- and warm-start Ford-Fulkerson, the first 5 images from the 30×30 HEAD image sequence

Image #	cold-start aug path #	cold-start aug path avg length	warm-start proj path #	warm-start proj path avg length	warm-start aug path #	warm-start aug path avg length
1	267	25.16	226	8.61	61	40.72
2	244	23.26	254	11.63	3	44.33
3	253	22.11	236	12.05	0	0
4	248	21.45	238	11.17	0	0
5	250	22.98	252	12.24	10	43.30

Table 11. Comparison of projection and augmenting paths in cold- and warm-start Ford-Fulkerson, the first 5 images from the 60×60 HEAD image sequence

Image #	cold-start aug path #	cold-start aug path avg length	warm-start proj path #	warm-start proj path avg length	warm-start aug path #	warm-start aug path avg length
1	789	46.52	674	10.36	164	56.57
2	852	44.17	763	9.69	99	62.59
3	752	41.09	866	14.71	0	0
4	782	40.19	567	7.52	169	48.0
5	777	42.62	931	16.67	0	0

Table 12. Comparison of projection and augmenting paths in cold- and warm-start Ford-Fulkerson, the first 5 images from the 30×30 SHOE image sequence

Image #	cold-start aug path #	cold-start aug path avg length	warm-start proj path #	warm-start proj path avg length	warm-start aug path #	warm-start aug path avg length
1	165	20.85	192	9.59	0	0
2	172	20.72	164	7.53	24	27.21
3	175	21.91	195	12.42	2	34.5
4	201	22.51	164	12.34	15	29.93
5	162	21.21	215	9.22	0	0

Table 13. Comparison of projection and augmenting paths in cold- and warm-start Ford-Fulkerson, the first 5 images from the 60×60 SHOE image sequence

Image #	cold-start aug path #	cold-start aug path avg length	warm-start proj path #	warm-start proj path avg length	warm-start aug path #	warm-start aug path avg length
1	585	41.22	580	13.40	31	58.65
2	508	40.21	562	14.06	0	0
3	609	42.29	469	7.13	147	50.65
4	646	43.86	675	14.13	17	58.24
5	595	44.64	683	14.82	0	0