arXiv:2303.00837v1 [cs.DS] 1 Mar 2023

用于加速 Ford-Fulkerson 算法的预测流

Sami Davies*、Benjamin Moseley†、Sergei Vassilvitskii‡、Yuyan Wang‡

2023年3月3日

摘要

近期研究表明，利用学习预测可以改善二分图匹配及类似组合问题算法的运行时间。在本文中，我们基于这一思想，通过使用预测流初始化 Ford-Fulkerson 算法，提升这一广泛用于计算最大流的算法的性能。我们提出的方法在预测质量方面展现出强大的理论性能。随后，我们研究图像分割这一计算机视觉中流的常见应用场景，并以强有力的实证结果补充理论分析。

1 引言

Ford-Fulkerson 算法是组合优化中理论与实践领域应用最广泛的方法之一。尽管它最初是为求解最大流问题而提出的，但调度（Ahuja 等人，1993）、计算机视觉（Vineet 和 Narayanan，2008）、资源分配、拟阵交（Im 等人，2021b）等许多领域的问题，都可以通过归约为流问题来求解。

从理论上说，存在渐近速度远快于原始 Ford-Fulkerson 算法的最大流算法；最新的成果是 Chen 等人（2022b）提出的近线性时间算法。然而，理论保证出色的算法往往难以在实践中实现，这种情况并不少见。事实上，实践中使用的算法仍有改进空间。实际上，在网络中计算最大流时，业内人士通常仍采用较早的算法，例如 Dinic 算法（Bhadra 等人，2020）、推送-重标记算法（Cherkassky 和 Goldberg，1997）、伪流算法（Chandran 和 Hochbaum，2009），或将这些算法与启发式方法结合，以适应特定场景。

在实际应用中，流算法通常用于解决随时间自然出现的多个问题实例。然而，理论分析以及许多实现都着眼于从头开始解决每个新问题，以获得最坏情况保证。这种做法不必要地丢弃了实例之间可能存在的信息。我们希望探究，能否通过利用以往示例中的信息，更高效地解决流问题。从一个非平凡的起点为算法提供初始状态，称为**热启动**。

我们的研究受到这样一个问题的启发：能否通过热启动 Ford-Fulkerson 算法来提升其理论和实证表现？为此，我们借助近期发展起来的带预测的算法框架（又称学习增强算法）。过去几年间的研究展现了利用学习到的预测来增强算法的强大作用，并由此推动了缓存（Im 等人，2022年； Lindermayr 和 Megow，2022年； Lykouris 和 Vassilvitskii，2021）、调度（Im 等人，2021a； Lattanzi

*西北大学。 sami@northwestern.edu。由 NSF计算创新奖学金 资助。

†卡内基梅隆大学。 moseleyb@andrew.cmu.edu。部分经费由 Google研究奖、Inform研究奖、Carnegie Bosch青年教职席位以及 NSF 资助项目 CCF-2121744 和 CCF-1845146 提供。

‡Google研究院——纽约。 sergeiv@google.com， wangyy@google.com。

等人，2020）、聚类（Lattanzi 等人，2021）、匹配（Chen 等人，2022a； Dinitz 等人，2021）等领域的发展（更多内容请参阅 Mitzenmacher 和 Vassilvitskii (2022年）综述）。如果一种算法能够利用传递问题实例相关信息的预测，就称其为**学习增强的**。以往的大多数研究都利用预测来克服在线场景中的不确定性。然而，Dinitz 等人(2021）近期开展的研究，以及 Chen 等人(2022a）的后续研究，则转而关注通过预测对偶变量，并利用这些变量热启动原始-对偶算法，从而缩短二分图匹配算法的运行时间。

受热启动组合优化算法这一理念的启发，我们希望通过热启动为流问题提供更快的运行时间保证。本文将重点讨论一般的流问题，同时也将展示计算机视觉中的一个常见实用场景：图像分割。在图像分割问题中，输入是一幅包含对象/前景的图像，目标是在图像中定位前景。

1.1 我们的贡献

对于配备容量向量 $c \in \mathbb{Z}_{\geq 0}^{|E|}$ 的图 $G=(V,E)$，设 f 是图 G 上的一个流，其中 f_e 是每条边 $e \in E$ 上的流值。设 $\mathcal{F}^*$ 是图 G 上所有可行最大流的集合。给定一个可能不可行的流 f，令 $\eta(f) = \min_{f^* \in \mathcal{F}^*} ||f - f^*||_1$。该项表示 f 距离最优状态有多近。

在一般网络上热启动 Ford-Fulkerson 算法 我们的主要贡献是算法 1，该算法可用于热启动任意一种 Ford-Fulkerson 实现，也就是说，可配合任意指定的增广路径搜索子程序使用。算法 1 以预测流 f作为输入。注意，f 可能对 G不可行，因为预测可能有误。算法 1 首先将 f 投影为 G上的可行流，然后从该可行状态运行 Ford-Fulkerson 过程，以求得最大流。虽然我们热启动的 Ford-Fulkerson 算法的性能取决于预测的质量，但它也具有与标准Ford-Fulkerson算法相同的最坏情况运行时间界。

定理 1。设 f 是网络 $G=(V, E)$上的一个可能不可行的流。设 T 为采用所选增广路径子程序时 *Ford-Fulkerson* 算法的最坏情况运行时间。使用相同的子程序，以 *1* 作为初始种子运行的算法，能够在 f时间内求得网络 f^*上的最优流 G，其时间为 $O(\min\{|E| \cdot \eta(f)\ T\})$。

在不同位置，我们会指定两种 Ford-Fulkerson 算法实现：Edmonds-Karp 算法和 Dinic 算法，其运行时间分别为 $T\ O(|E|^2|V|)$ 和 $O(|V|^2|E|)$。

人们可能会想知道如何获得这样的 f。我们证明，当网络来自一个固定但未知的分布时，可以对最优流的最佳近似进行 PAC 学习（定理 10），并将其用于热启动。

在局部变化的网络上实现更快的热启动 接下来，我们改进算法1在网络实例上的分析，这些实例彼此之间存在渐进的局部转变。我们证明，网络之间的局部转变（定理 2 的非正式表述）会产生许多可用于发送流的短路径，从而缩短运行时间。

定理 2（非正式表述，正式表述见定理 14）。固定可分网络 G^1 和 G^2，它们之间的转变是 d-局部的。对于 $\widehat{f}$ 上的一个最优流，算法 *1* 以 $\widehat{f}$ 在 G^2 上启动，以求得 f^* 在 G_2 上的最优流，其运行时间为 $O(d \cdot |E| + d^2 \cdot \eta(\widehat{f}))$。

实证结果 受理论结果启发，我们将热启动的 Ford-Fulkerson 算法应用于由以下实例构建的网络： 图像分割 这些实例涉及对运动物体在不同角度下拍摄的照片序列进行分割。我们证明，热启动比标准 Ford-Fulkerson 算法（运行时间提升 2-5× 倍）更快，从而表明我们的理论能够预测实际表现。热启动带来的速度提升，关键之处并非在于沿更少的路径发送流，而是在于使用更短的路径，将 f 投射到可行流上，正如定理2所预测的那样。我们注意到，

我们实验的目标不一定是为图像分割提供最先进的算法，而是展示与从头运行 Ford-Fulkerson 算法相比，热启动 Ford-Fulkerson 算法能在实际网络上显著缩短运行时间。

1.2 相关工作

流问题已得到充分研究。可参阅Ahuja 等人 (1993)的综述。Ford-Fulkerson 算法通过反复调用寻找增广路径的子程序，以贪心方式计算最大流（Ford 和 Fulkerson，1956）。不同的子程序催生了不同的实现，例如 Edmonds-Karp 算法（使用广度优先搜索）（Edmonds 和 Karp，1972）以及更快的 Dinic 算法（Dinitz，2006年）。Sherman(2013)和Kelner 等人 (2014)提出了能够快速计算近似最大流的算法。Chen 等人(2022b)提出了一种近乎线性时间的最大流算法。

与我们的工作思路相近，Altner 和 Ergun (2008)通过实验证明，可以在相似网络上对推送-重标记算法进行热启动。此外，我们还注意到同期有一项关于热启动最大流算法的工作，作者为Polak 和 Zub (2022年)。重要的是，他们要求预测流满足流守恒约束；作者也指出了这一限制。相比之下，我们明确设置了可行性恢复步骤，因此无需这一假设。

学习增强算法近年来逐渐流行起来。这一领域由Kraska 等人 （2018）的工作开启，他们展示了学习型数据结构方面的成果。此后，这一领域在在线算法设计中日益流行，算法利用预测来应对不确定性（Purohit 等人，2018；Lattanzi 等人，2020年；Antoniadis 等人，2020）。对于不太熟悉这方面文献的读者，我们建议阅读第*1.3*节中的学习增强算法段落。

虽然Kraska 等人 （2018）表明可以利用预测改善运行时间，但从理论上讲，这仍未得到充分理解。Dinitz 等人 （2021）的工作展示了如何改善匈牙利算法在带权二分图匹配中的运行时间。Chen 等人（2022a）将这一方法扩展到了其他图问题。这两项工作都利用预测得到的对偶解，对原始-对偶算法进行热启动。此外，预测也被用于改善其他算法的运行时间。例如，Sakaue 和 Oki （2022年）提出了用于加速离散凸分析的算法，而Lu 等人 （2021）则表明，预测可以用于改善广义排序的运行时间。一个密切相关的领域是数据驱动的算法设计（Gupta 和 Roughgarden，2017；Balcan 等人，2021）。

1.3 组织与预备知识

第 2 节介绍热启动算法，证明其正确性，并给出运行时间保证。在第 3 节中，如果网络属于特定子类，我们会给出额外的理论保证。在第4节中，我们展示了图像分割所产生网络上的实证结果，这些网络与第 3 节中的网络密切相关。更多实证证据见附录 6 ，更多内容请访问
https://github.com/wang-yuyan/warmstart-graphcut-algorithms-pulic。

学习增强算法 在这种模型中，算法可以使用一个预测参数。预测可能有误，因此必须配有误差度量。在我们的设定中，对于给定的网络 G，我们将预测一个整数流 f。该预测流对于 G 可能不可行。回想一下，对于 $\mathcal{F}^*$ 上最优流的集合，我们将 f 在 G 上的误差定义为 $\eta(f) = \min_{f^* \in \mathcal{F}^*} ||f - f^*||_1$。

学习增强算法领域的研究已充分确立，预测和算法所需具备的属性是可学习性、一致性和鲁棒性。我们证明，在对最优流的唯一性作出额外假设的情况下（见假设8），预测流是 PAC 可学习的（定理10）。此外，我们利用流的实例鲁棒性(Lavastida 等人，2021) 来论证特殊网络的可学习性（定理2 ），并通过实证观察到这一点。如果给定一个预测流 f，它适用于

网络 G，且该预测流实际上是最优流，那么运行时间为 0，因此算法1 具有一致性。我们还^能够保证鲁棒性和最坏情况保证，因为算法1的运行时间为 $O(\min\{|E|\eta(f)\ T\})$，^它会随着 f 与网络上的最优流 f^* 相差程度的变化而平滑退化，但最坏情况下仍受 $O(T)$[1]限制。

网络流 设 $G=(V,E)$ 是一个固定网络，其中包含 $|V|=n$ 和 $|E|=m$。源点 s 和汇点 t属于顶点集 V。该网络配有容量向量 $c\in\mathbb{Z}^m_{\geq 0}$。若流 $f\in\mathbb{Z}^m_{\geq 0}$ 在G 上满足流守恒，即对于所有顶点 $u\in V\setminus\{s,t\}$，流入 u 的流量等于流出$\sum_{e=(v,u)} f_e = \sum_{e=(u,w)} f_e$的流量；并满足容量约束，即 $f_e\leq c_e$ 对所有边 $e\in E$均成立，则称该流可行。本文将满足这些约束的流称为**可行**流。

给定一个满足容量约束但未必满足流守恒的流 f，其**残量图** G_f 是在相同顶点集上构造的网络。对于任意边 $e=(u,v)\in E$，添加一条从 e 到 G_f、容量为 $c'_e=c_e-f_e$的边，以及一条反向边$e=(v,u)$，其容量为 f_e。设 $\nu(f)$ 为 f 从源点发送的流量，$\nu(f)=\sum_{e=(s,u)} f_e$。$G_f$ 中的一条**增广路径** 是从 s 到 t的一条路径 p，其中每条边 $e\in p$ 都具有 $c'_e>0$。

当 f 不满足流守恒时，某个节点的总流入量与总流出量不同。如果流入量大于/小于流出量，则分别称该节点的差值为**正盈余/亏空**（$\mathrm{ex}_f/\mathrm{def}f$）。为简洁起见，我们将 G中根据流 f计算的总盈余和总亏空分别记为 $\mathrm{ex}_f=\sum_{u\notin\{s,t\}}\mathrm{ex}f(u)$ 和 $\mathrm{def}_f=\sum_{u\notin\{s,t\}}\mathrm{def}f(u)$；注意，这不包括源点和汇点。令$A_f,B_f\subseteq V$分别表示相对于 f具有正盈余/亏空的节点集合（因此 $t\in A_f$, $s\in B_f$）。进一步将这些集合分别去除 s,t 后得到的集合记为 $A'_f=A_f\setminus\{t\}$ 和 $B'_f=B_f\setminus\{s\}$，这样会更方便。

2 热启动 Ford-Fulkerson

这里给出一种利用预测流的算法。下一个命题源于以下观察：任何 Ford-Fulkerson 方法（例如 Edmonds-Karp 算法或 Dinic 算法）都可以使用任意可行流进行初始化，因为以可行流为初始流运行 Ford-Fulkerson 算法，与从头开始在残量图上运行该算法是相同的。Ford-Fulkerson 算法的每次迭代都会增加流的值，并需要 $O(|E|)$ 的时间来寻找一条增广路径并发送流。

命题 3。令 f为网络 G上的一个可行流，其中 $\nu(f)<\nu(f^*)$ 表示网络 G上的一个最优流。以 f为初始流的 *Ford-Fulkerson* 算法至多经过 $\nu(f^*)-\nu(f)$ 次迭代即可终止，因此其运行时间为 $O(|E|\cdot(\nu(f^*)-\nu(f)))$。

^令 f 为网络 G上的预测流。该流可能不可行，也就是说，它可能违反容量约束或流^守恒约束。算法 1 包含两个主要步骤：步骤（1）将 f 投影为可行流 —— 我们将其称为**可行性投影** —— 步骤（2）使用可行流作为初始流运行 Ford-Fulkerson 方法，并找到一个最优流。在可行性投影过程中，我们首先在违反容量约束的地方向下取整流。然后沿着**投影路径**发送流，即沿着从盈余节点到亏空节点的路径发送流，其中残量图上的所有容量均为正值，从而实现流守恒。

设 f 是一个满足容量约束且为整数的不可行流。在算法 1 的主 WHILE 循环中，分三个回合寻找投影路径：$A'_f-B'_f$、A'_f-s、B'_f-t。在每个回合中，我们通过构造辅助图 G'，并在该图上应用 Ford-Fulkerson 算法中选定的增广路径子程序（例如广度优先搜索），来寻找投影路径。为构造 G' 以寻找所有可能的 $A'_f-B'_f$ 投影路径，取关于 f 的残量图 G_f，并将其视为一个新网络。添加一个超级源点 s^*和一个超级汇点 t^*。为每个 $u\in A'_f$（非汇点盈余节点）添加弧 (s^*,u)，其容量为 $\mathrm{ex}f(u)$；并从每个 $u\in B'_f$（非源点亏空节点）添加弧 (u,t^*) 至 t^*。将该网络上的所有流初始化为 0。一个 s^*-t^*

[1]请记住，T 取决于 Ford-Fulkerson 实现。

算法 1 热启动 Ford-Fulkerson 算法，使用 $\widehat{f}$

输入：预测流 $\widehat{f}$
当 ∃ 边 e 属于 G 且 $\widehat{f}_e > c_e$ **时**
  更新 $\widehat{f}$ to $\widehat{f}_e \leftarrow c_e$
设置 $f \leftarrow \widehat{f}$
构建残量图 G_f，以及 A'_f和 B'_f，即要进行舍入的超额/亏空节点集合
//主 *WHILE* 循环，可行性投影
当 $|A'_f \cup B'_f| > 0$ **时执行**
  如果 $|A'_f| > 0$ **则**
    如果 ∃ 从 $u \in A'_f$ 到 $v \in B'_f$ 的投影路径**则**
      令 p 为从 u 到 v的路径
    else
      设 p 是一条从 $u \in A'_f$ 到 s 的路径
  else
    设 $v \in B'_f$，设 p 是一条从 t 到 v 的路径
  //有关路径搜索的更多详情，请参阅正文
  设 w，w' 分别是 p 的起始节点和终止节点
  沿路径 p 发送 $\mu_p = \min\{\text{ex}_f(w), \text{def}_f(w'), \min_{e \in p} c'_e\}$ 个单位的流
  更新 f、G_f，A'_f 和 B'_f
//可行性投影结束，优化开始
以 G 为种子运行 Ford-Fulkerson 算法，使用 f，直至达到最优性
输出：f^*

在 G' 中的增广路径对应于 G_f 中的一条 $A'_f - B'_f$ 投影路径。这是因为，对于从 u 到 u' 的任意投影路径，可以令 s^* 到 u 以及 u' 到 t^* 上存在流，从而使其成为 G' 中的一条增广路径，反向过程也成立。其余两个回合的 G' 图也以类似方式构造；为寻找$A'_f - s$ 投影路径，向 $u \in A'_f$ 添加弧 (s^*, u) 和 (s, t^*)；对于 $t - B'_f$，为每个$v \in B'_f$添加弧 (s^*, t) 和 (v, t^*)。

要说明在算法 1 的主 WHILE 循环中，可以先从 $A'_f - B'_f$，然后从 $A'_f - s$，最后从 $B'_f - t$找到投影路径，我们使用以下引理。

引理 4。固定一个不可行流 f。如果在 G_f 中不存在一条从 A'_f到 B'_f具有正容量的路径，那么发送流量从 A'_f 到 s （或从 t 到 B'_f）以形成流 h，不会导致在 G_h中出现一条从 A'_h 到 B'_h 具有正容量的路径。

证明。为导出矛盾，假设（不失一般性）从某个$u \in A'_f$ 向 s 发送流形成了流 h，且在 G_h中存在一条从 A'_f 到 B'_f 的正容量路径。设$p_1 = (u_1, \ldots, s)$为在 G_f中发送流所经过的路径，设 $p_2 = (u_2, \ldots, v)$为 G_h中具有正容量的路径。根据假设，注意到 u_1、$u_2 \in A'_f$ 和 $v \in B'_f$。由于 p_2 在 G_f中的容量从 0 变为在 G_h 中的正值，因此 p_2中至少有一条边 $e = (a, b)$在 p_1中具有$e = (b, a)$。若存在多条这样的边，则令 e为 p_1. 中最后一条这样的边。令 $p'_1 = (u_1, \ldots, a)$为 p_1 在a 处的截断，令 $p'_2 = (a, \ldots, v)$为 p_2 中从 a开始的后缀。然后令 p' 为 p'_1 与 p'_2的拼接。注意到 p'_1 和 p'_2 在 G_f中都具有正容量。因此，p' 是 G_f 中一条从u 到 v且具有正容量的路径，这与假设矛盾。

□

2.1 热启动算法分析

本节将分析算法 1 的工作原理。

算法的有效性 我们将证明，在主 WHILE 循环中必定存在一条投影路径。

引理 5. 给定不可行流 f, $\forall u \in A'_f$, $\exists v \in B_f$，使得存在一条从 u到 v的投影路径；$\forall v \in B'_f$, $\exists u \in A_f$，使得存在一条从 u 到 v的投影路径。

该引理源于以下观察：对于任意固定的节点集合，超额量/亏缺量之和与流入流量和流出流量之差相关。

命题6. 设 f 是满足网络 G的容量约束的流。那么，对于任意 $S \subseteq V$， S 中的总亏缺量与 S 中的总超额量之差，恰好等于流出 S 和流入 S的总流量之差。形式化地说，

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

证明。 在 $\sum_{u \in S} \mathrm{def}_f(u) - \sum_{u \in S} \mathrm{ex}_f(u)$ 中，流量处于 S内的边会分别以正号和负号计数，而流入 S 或流出 S 的边则分别在超额量和亏缺量中各计数一次。

□

引理 5 的证明。 我们通过反证法证明一个方向。另一个方向的证明类似。对于任意 $u \in A'_f$，假设不存在这样的路径。在命题 6 中，取 S 为从 u 在 G_f中可到达的所有顶点的集合。 S 中的节点都不可能具有正的亏空，因此命题 6 的左侧必为负数。另一方面， S 的流入量必须为 0，否则就存在一条从 S 指向 $V \setminus S$ 中 G_f的边，从而产生一个 $V \setminus S$ 中可从 u 到达的顶点，这与 S的极大性矛盾。因此，命题 6 的右侧为非负数，这与该等式矛盾。

□

请注意，每次迭代都会使系统中的超额和亏空总量（即 $\mathrm{ex}_f + \mathrm{def}_f$）至少减少一个，因此 WHILE 循环会在恢复流守恒后终止，从而得到一个可行流。随后，标准Ford-Fulkerson算法继续运行，直到找到最优解。

运行时间分析 这里我们着手证明定理 1。我们证明运行时间与$\widehat{\ }$ $\widehat{\ }$预测质量 $\eta(f) = \min_{f^* \in \mathcal{F}^*} ||f - f^*||_1$有关。我们首先界定寻路子程序 $\widehat{\ }$的调用次数。对于投影路径，总超额和亏空最多可能增加 $\sum_e \max\{f_e - c_e, 0\}$，这是因为算法 1 会将超过容量的流向下取整。因此，最多需要 $(\mathrm{ex}_{\widehat{f}} + \mathrm{def}_{\widehat{f}})$ 次投影路径来恢复可行性。对于增广路径，流值之差 $\nu(\widehat{f}) - \nu(f^*)$ 最多可能减少 $\sum_e \max\{f_e - c_e, 0\}$，在向下取整过程中最多减少 $\mathrm{ex}_{\widehat{f}} + \mathrm{def}_{\widehat{f}}$，在可行性投影过程中最多再减少这么多，因此总次数最多为这三者之和。每次寻路需要 $O(|E|)$ 的时间。结合下面的引理即可证明定理1。

$\widehat{\ }$**引理 7.** $\nu(f^*) - \nu(f)$，例如 $\widehat{f} + \mathit{def}_{\widehat{f}}$，以及 $\sum_e \max\{f_e - c_e, 0\}$ 均有上界 $\eta(f)$。

$\widehat{\ }$证明。首先，我们看到 $|\nu(f) - \nu(f^*)|$ 与 $|\nu(f) - \nu(f^*)|$ 之间的差异，只可能来自投影 f 到可行流 f. 过程中损失的值，以及增加的超额和亏空$\widehat{\ }$。因此，我们可以为 $|\nu(f) - \nu(f^*)|$ 给出上界：

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

然后，我们可以进一步 $\widehat{}$上界 $|\nu(f) - \nu(f^*)|$ 通过重写流值之间的差 ws

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

$\widehat{}$接下来要用 ℓ_1 误差来界定的项是 $\sum_e \max\{f_e - c_e, 0\}$，不过很容易看出

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

$\widehat{}$最后，我们看到，任意节点 $v \in V \setminus \{s,t\}$ 的超额/亏空都可以归因于 f_e与 f^*_e之间的差值，其中 e与 v相邻，因为任意 $f^* \in \mathcal{F}^*$在所有非源点和汇点节点上的超额/亏空均为 0。因此，$\widehat{}\mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}} \leq \eta(f)$。

□

$\widehat{}$定理证明 *1*。给定一个流 f，它不满足容量约束，算法 1 只需更新$\widehat{}$ $\widehat{}$边 $E' \subseteq E$，将违反容量约束的边 $f_e > c_e$ 设为 $f_e \leftarrow c_e$，对于 $e \in E'$。这可以在 $O(|E'|$时间内完成。此外，将这些边上的流向下取整，会使流值以及超额与$\widehat{}$亏空之和最多变化 $\sum_e \max\{f_e - c_e, 0\}$。

在引理5中，我们证明了给定 $\widehat{}$
f 满足容量约束，算法1 将找到一个最优f^*。接下来，我们分析算法1的运行时间。算法1 中主 WHILE 循环的每次迭代耗时 $O(|E|)$。此外，算法1中主 WHILE 循环的迭代次数至多为$\mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}}$。令 f 为算法1 在主 WHILE 循环结束时得到的可行流。生成流 f的 $O(|E|(\mathsf{ex}_{\widehat{f}} + \mathsf{def}_{\widehat{f}}))$所需的运行时间为。

最多需要 $|\nu(f^*) - \nu(f)|$ 次 Ford-Fulkerson 算法迭代，才能根据命题 $\nu(f^*)$ 从 f 达到最优流值3。Ford-Fulkerson 算法的每次迭代也需要 $O(|E|)$的时间。

因此，算法的运行时 间为：算法1 在给定满足容量约束的预测时，至多需要 t

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

将其与投影以满足容量约束所产生的损失相结合，算法1 的完整运行时间为

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

$\widehat{}$我们证明了上式中乘以 $|E|$ 的所有项都可以界定为 $O(\eta(f))$（引理 7）。

不难证明，算法 1 的最坏情况运行时间有上界 $O(T)$。在可行性投影步骤中，只需构造三次辅助图（回顾引理4，我们可以分别沿 $A'_f - B'_f$条路径、再沿 $A'_f - s$ 条路径、最后沿 $t - B'_f$条路径发送流），每次构造的图都包含 $|V| + 2$ 个顶点和$O(|E|)$ 条边。因此，在这些图上运行选定的 Ford-Fulkerson 实现需要 $O(T)$ 的时间。回顾一下，优化步骤同样需要 $O(T)$ 的时间，因为从一个可行流开始运行 Ford-Fulkerson，等价于将残量图作为新输入从头运行该算法。因此，总运行时间为 $O(T)$。

□

2.2 流的 PAC 学习

在这里，我们从理论上证明，高质量的流是可学习的，从而证明可以学习流，作为算法 1的输入。我们证明，给定网络容量向量上的一个分布，可以从样本中学习到一个预测流，且该预测流是最佳近似。

考虑一个固定的网络 G，其边容量 c为整数。一个实例是一个网络 G^i，它与 G具有相同的顶点集和边集，但容量向量为 c^i，其中 G^i中的每条边 e都必须满足 $c_e^i \in [0, c_e] \cap \mathbb{Z}$。设 $\mathcal{D}$ 为此类实例上的一个未知分布。由于一个实例完全由其新的容量向量确定，我们将其记作对容量向量 $c^i \sim \mathcal{D}$进行采样。

假设我们从 $\mathcal{D}$ 中抽取实例 c^1、.....、c^s。令 $\mathcal{F}$ 为 G 上满足 c 中容量约束的所有整数流的集合。需要注意的是，$\mathcal{F}$ 中的流不必满足流守恒。严格来说，一个网络G 可能有多个最优解。这里我们作出以下假设。

假设 8。对于网络 G，存在唯一对应且可计算的最优流。[2]

给定样本 c^1、.....、c^s，我们可以计算样本 $f^*(c^1)$、.....，$f^*(c^s)$ 上唯一对应的最优流。令 f 表示一个预测流。当我们的目标是对 Ford-Fulkerson 算法进行热启动时，我们选择 $\mathcal{F}$中使经验风险 $\widehat{f} = \operatorname{argmin}_{f \in \mathcal{F}} \frac{1}{s} \sum_{j=1}^{s} ||\widehat{f} - f^*(c^j)||_1$. 最小的预测流。根据假设 8，如引理 9所示，该预测流可以高效地计算出来。

引理 9。通过对所有 $e \in E$ 取 $\widehat{f}_e =$ 中位数$(f^*(c^1)_e$、...、$f^*(c^s)_e)$，可以在多项式时间内根据独立样本c^1、...、$c^s \sim \mathcal{D}$找到使 $\frac{1}{s} \sum_{j=1}^{s} ||\widehat{f} - f^*(c^j)||_1$ 最小的流 $\widehat{f} \in \mathcal{F}$。

证明。我们希望找到使 $\frac{1}{s} \sum_{j=1}^{s} ||\widehat{f} - f^*(c^j)||_1$. 最小的$\widehat{f} \in \mathcal{F}$。由于我们不要求流守恒，因此可以分别针对每条边进行最小化，其中 f_e 将属于 $[0, c_e] \cap \mathbb{Z}$，也就是说，只需针对每个 $e \in E$最小化 $\frac{1}{s} \sum_{j=1}^{s} |\widehat{f}_e - f^*(c^j)_e|$。函数 $\frac{1}{s} \sum_{j=1}^{s} |\widehat{f}_e - f^*(c^j)_e|$ 关于 $\widehat{f}_e$是连续且分段线性的，其斜率在点 $\{f^*(c^j)_e\}_j$处发生变化。众所周知，该函数在 $[0, c_e]$中的最小值是 $f^*(c^1)_e$、...、$f^*(c^s)_e$ 的中位数。

□

现在我们将陈述 我们的 PAC 学习结果。该定理的证明沿用了 Dinitz 等人 (2021) .

定理 10. 令 *et* $c^1, \ldots, c^s$ 从 $\mathcal{D}$ 中独立同分布采样，并令 $\widehat{f} = \textit{argmin}_{f \in \mathcal{F}_0} \frac{1}{s} \sum_{j=1}^{s} ||f - f^*(c_j)||_1$. *Fo* r

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

且 $\widetilde{f} = \textit{argmin}_{f \in \mathcal{F}_0} \mathbb{E}_{c^i \sim \mathcal{D}} ||f - f^*(c^i)||_1$，则以至少 $1 - p$ 的概率，$\widehat{f}$ 满足 s

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

在定理 10 的证明中，我们将使用一些关于函数类伪维度的著名结果。

VC 维数是用于衡量一族二值函数复杂度的量，而伪维度则是它在实值函数中的对应概念。具体而言，一族实值函数 $\mathcal{H}$ 的**伪维度** 是被 $\mathcal{H}$打散的最大子集的大小。如果存在实值见证值 r_1，$\cdots$，r_s，使得对于 2^s 的每个子集 T，都存在一个函数 $h \in \mathcal{H}$ 满足 $h(x_i) \leq r_i$ 当且仅当 $i \in T$，那么 X的子集 $S = \{x_1, \ldots, x_s\}$ 就被 $\mathcal{H}$ **打散**。

以下定理描述了某些 $h \in \mathcal{H}$ 的样本均值如何收敛到其期望，这种关系取决于伪维度。

定理 11 (一致收敛)。设 $\mathcal{H}$ 是一个定义域为 X、值域在 $[0, H]$ 中的函数类。设 $d_{\mathcal{H}}$ 为 $\mathcal{H}$ 的伪维度。对于任意在 X 上的分布 $\mathcal{D}$、任意 $\epsilon > 0$ 以及任意 $\delta \in (0, 1]$, 若

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

[2]这类假设是基于学习增强的运行时间改进中 PAC 学习结果的标准假设，即使没有明确说明也是如此。例如，请参阅 Sakaue 和 Oki (2022年)。

对于某个常数 c，则在 s 个样本 $x_1 \ldots$、$x_s \in \mathcal{D}$ 上，以至少 $1-\delta$ 的概率，

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

有了定理 11，我们现在可以证明 PAC 学习结果了。

定理 *10*的证明。我们将构造一个函数类，其中包含流 f在容量约束 c^i下对应的损失函数。然后，我们将定理 11 应用于这个函数类。

对于每个满足容量向量 c^i 的整数流 $f \in \mathbb{R}^{|E|}$，令函数 $g_f(c^i) = ||f^*(c^i) - f||_1$ 为 f 在 c^i 上的损失函数。然后，令 $\mathcal{H} = \{g_f \mid f \in \mathbb{R}^m\}$ 为所有这些损失函数构成的族。

我们在引理 9 中看到如何高效地计算经验风险最小化器。此外，损失函数值域的上界，即定理 H 的陈述中 11所述的上界，至多为 $m \cdot \max_e c_{e^\circ}$ 要证明我们的引理，还需给出 $\mathcal{H}$的伪维度上界。

我们将通过证明 $\mathcal{H}$ 的伪维度不超过另一个函数类 $\mathcal{H}_m$的伪维度，来给出 $\mathcal{H}_m = \{h_y \mid y \in \mathbb{R}^m\}$ 的$h_y(x) = ||y - x||_1$的伪维度上界；后者的伪维度已知。以下结果作为定理 19 出现在 Dinitz 等人 (2021) 的论文中，读者可参阅该论文了解其证明。

引理 12。 $\mathcal{H}_m$ 的伪维度至多为 $O(m \log m)$。

现在只需证明下面的引理，该引理联系了两个类的伪维度。

引理 13。如果 *t he* $\mathcal{H}_m$ 的伪维度至多为 d，那么 $\mathcal{H}$ 的伪维度至多为 d .

证明。设 $S = \{c^1, \ldots, c^d\}$ 是被 $\mathcal{H}$ 打散的集合。设 r_1，……，$r_d \in \mathbb{R}$ 是相应的见证值，使得对于所有 $S' \subseteq [d]$，存在某个 $g_{f^{S'}} \in \mathcal{H}$，满足 $g_{f^{S'}}(c^j) = ||f^*(c^j) - f^{S'}||_1 \leq r_j$ 当且仅当 $j \in S'$。

我们将从 S 构造一个大小为 d 的集合 $\widetilde{S}$，该集合被 $\mathcal{H}_m$ 打散。令 $\widetilde{S} = \{f^*(c^1)$ ……，$f^*(c^d)\}$，并固定某个 $S' \subseteq [d]$。那么 $h_{f^{S'}}(f^*(c^j)) = ||f^{S'} - f^*(c^j)||_1 \leq r_j$ 当且仅当 $j \in S'$。 □

代入 $H \leq m \cdot$最大值$_e\, c_e$和 $d_{\mathcal{H}} \leq O(m$对数 $m)$ 到定理 11中，可知只需取

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

□

3 通过更短的投影路径实现更快的流

这里，我们证明，对于某一类网络，算法 1 还能进一步加速。直观而言，额外的提速来自找到更短的投影路径。随后，我们将在第 4 节说明简单的图像分割网络如何属于这一类，因此这一理论解释了我们在图像分割实例上观察到的提速。

设 $G = (V, E)$ 是一个有向图，其中有 s，$t \in V$。假设 $V \setminus \{s, t\}$ 构成一个二维网格。此外，对于 u，$v \in V \setminus \{s, t\}$，若 $e = (u, v) \in E$，则反向边 $e = (v, u) \in E$。我们考虑定义在 G 上的一对网络 G^1，G^2。这两个网络的唯一区别在于其容量向量，不过我们假设它们的容量向量为 c^1，$c^2 \in \{1, M\}^m$，其中 M 是某个较大的整数，并假设所有与 s 或 t 相 incident 的边的容量均为 M。

对于 $\ell \in [2]$，令 $E_\ell = \{e \in E \mid c_e^\ell = 1\}$。如果 $V \setminus \{s, t\}$ 中的顶点可以划分为子集 V_ℓ和 $W_\ell = V \setminus (\{s, t\} \cup V_\ell)$，使得存在某个 $x \in V_1 \cap V_2$，满足（x，t）$\in E$，且存在某个$y \in W_1 \cap W_2$，满足（s，y）$\in E$，并且对于所有 $e =$（u，v）$\in E_\ell$，e的一个端点在 V_ℓ中，另一个端点在 W_ℓ中，则称 G^ℓ **可分离**。我们

称 G^1 与 G^2 之间的转移是**d-局部**的，如果对于任意一对不同的节点 u，$v \in (V_2 \setminus V_1) \cup (W_2 \setminus W_1)$，它们之间的距离（即两者间最短路径的长度）至多为 d。这里，d 控制投影路径的长度。

虽然我们的理论结果需要额外的假设，但第 4 节中的实证结果表明，这些假设对于算法 1利用较短的投影路径来说是充分条件，但并非必要条件。

对于定理 2 的可行性投影证明，$\widehat{f}$ 包含几个步骤。首先，算法1 可以选择沿其传输流的路径，使得超额和亏空要么得到修正，要么 V_2 仅包含具有正盈余的节点，而 W_2 仅包含具有正亏空的节点，反之亦然。我们论证 $\widehat{f}$ 至多通过长度为 $\eta(f)$ 的路径修正总超额和亏空 $O(d)$。接下来，仍需使用 V_2 和 W_2 来修正 s 和 t中的超额/亏空。这些路径的长度无界，即长度为 $O(|E|)$，并且我们可以论证流值的变化至多为 $||E_2| - |E_1|| \leq O(d^2)$，其中上界来自 d-局部性的定义。

定理 14（重述定理2）。*固定可分网络 G^1 和 G^2，它们之间的转变是 d-局部的。对于 $\widehat{f}$ 在 G^1上的一个最优流，算法 1 以 $\widehat{f}$ 在 G^2 上为求得一个最优流f^*的运行时间为 $O(d^2 \cdot |E| + d \cdot \eta(\widehat{f}))$。*

证明。对于具有容量约束 c^2 的网络 G^2，按照算法 1 将 $\widehat{f}$ 投影为满足 c^2，并令所得流为 f。注意，只有当某条边上的流在从 f 到 f 的投影过程中被向下取整时，与该边相连的节点才可能位于 $A'_f \cup B'_f$ 中。只有当边 (u, v) 从一条两端的 u 和 v 要么都在 V_1 内、要么都在 V_1 外的边变成边界边时，其流才会在投影过程中被向下取整；也就是说，u 或 v 中有一个位于 V_2 内，另一个位于 W_2 内。

将 f 投影为满足流守恒的流，算法 1可以选择路径，先将流从 V_2 中具有正盈余的节点路由至 V_2中具有正亏空的节点，再将流从 W_2 中具有正盈余的节点路由至 W_2中具有正亏空的节点。根据我们对网络为 d–局部网络的假设，V_2 或 W_2 中的盈余和亏空之间的距离至多为 d。由于非边界边的容量为 M，算法 1 可以沿这些投影路径路由流，直到 V_2 和W_2 中分别只剩下盈余或亏空。此外，如果 V_2 中有一个节点存在正亏空，且 W_2 中有一个节点存在正盈余，并且存在从 V_2 到 W_2的正流量边，则可以沿反向边发送流，并通过长度至多为 $2d$的投影路径消除该盈余/亏空。

此外，算法 1 可以选择路径来执行流路由，使盈余和亏空在边界两侧呈对称分布。具体而言，对于节点 $u \in V_2$，可以路由流，使 $\mathsf{ex}_f(u) = \sum_{v:(u,v)\in E_2} \mathsf{def}_f(v)$，亏空也同样如此；对于 W_2 中的节点亦可作类似处理。注意，由于$\sim M$ 足够大，因此可以进行这种重新路由。根据定理 1 的证明，我们知道 $\mathsf{ex}_f + \mathsf{def}_f \leq \eta(f)$，因此重新路由这些盈余和亏空的运行时间至多为 $O(d \cdot \eta(f))$。

我们将证明，任何剩余的亏空/超额都可以通过使用 s 和 t的路径来处理。首先，G^1 上的最大流值为 $|E_1|$，因为从 W_1 进入 V_1 的边构成一个割。另一方面，对于较大的 M，并且由于 $x \in V_1 \cap V_2$ 和 $y \in W_1 \cap W_2$的存在，存在一些可行流，使得每条边（u，v）的流量为 1，且流入 V_1的边包括$u \in W_1$ 和 $v \in V_1$。其次，我们的重新路由过程使V_2 内的超额与边界另一侧 W_2中的亏空对称，因此，要么 V_2 中存在正盈余且 W_2中存在正亏空，要么反之。

我们使用这两个观察结果。假设重新路由后，V_2 内部存在亏空，而其外部存在超额。因此，根据我们的路由方式，可以假设 E_2 中所有具有正流量的边都从 W_2 指向 V_2。对于 W_2中每个具有正盈余的节点 u，取出 u 的超额并将其发送到 s；根据可分离的条件，这是可行的。同样，对于 V_2内部每个具有正亏空的节点 u，找到从 t 到 u 的路径，并将 $\mathsf{def}_f(u)$ 从 t 发送到 u；根据可分条件，这同样可行。由此得到的

流是可行的。此外，不存在 $s-t$ 路径，因为 E_2 中所有进入 V_2 的边都已饱和并构成一个割，因此该流是最优的。所以，使用 s 和 t 对这个流进行重新路由，耗时为 $O(|E|(|E_1|-|E_2|))$。

重新路由后，如果 V_2 内有超额而 W_2 内有亏空，证明也类似。没有超额/亏空与之相连的边，其流量为 1，流入 V_2。由于亏空位于 V_2 外部，穿过从 V_2 到 W_2 的边界边，其流量为 1，流出 V_2。将流从 s 发送到 W_2 内亏空为正的节点，并将 V_2 内的超额发送到 t；根据可分离的条件，这是可行的。所得的流是可行流，但可能并非最优，因为可能需要饱和新的边界边。运行时间仍为 $O(|E|(|E_2|-|E_1|))$。

根据 d–局部的定义，$||E_2|-|E_1|| \leq O(d^2)$。因此，算法 1 在这些局部变更后的实例上的运行时间至多为 $O(d^2 \cdot |E| + d \cdot \eta(\widehat{f}))$。□

4 实证结果

在本节中，我们验证第 2 节和 3[3]中的理论结果。我们研究图像分割，这是计算机视觉中的一个核心问题，旨在将给定图像中的对象与背景分离。一系列研究将该问题重新表述为最大流/最小割优化问题（Boykov 和 Jolly，2001年；Boykov 和 Kolmogorov，2004年；Boykov 和 Funka-Lea，2006年），并使用组合图割算法求解，其中包括 Ford-Fulkerson 算法。

我们并不试图在图像分割方面提供最先进的运行时间结果。我们的目标是展示，在真实世界网络上，与冷启动 Ford-Fulkerson 相比，热启动 Ford-Fulkerson 算法能够大幅缩短运行时间。我们重点介绍以下几点：

- 对于 Edmonds-Karp 和 Dinic 算法实现的 Ford-Fulkerson 算法，热启动相比从头开始运行算法（称为**冷启动**）能够缩短运行时间。
- 随着图像像素数量（即分辨率）的增加，构造的图规模也会增大，节省的时间也会更加显著。
- 算法 1 中的可行性投影步骤性能出色。它返回的可行流仅略微次优，并且会寻找短路径来修正超额和亏空。这两个因素都使热启动比冷启动高效得多。

数据集与数据预处理 我们使用弗赖堡大学模式识别与图像处理 数据集中的四个不同图像组[4]，，分别名为 BIRDHOUSE、HEAD、SHOE 和 DOG。前三个组来自图像序列[5]，数据集，格式为 .jpg图像；而 DOG来自立体视觉自运动数据集[6]，，是一个视频，我们将其转换成了 .jpg 格式。

每个图像组都包含一个序列，其中的照片展示同一个对象和背景。序列可能呈现对象相对于背景的运动，也可能呈现相机拍摄角度的变化。序列中的每张图像都只与前一张略有不同，这可能导致分割解出现细微差异。因此，可以使用前一张图像对应最大流问题的最优流进行热启动。

我们从每个组中选取 10 张图像，并将其裁剪为 600×600 像素，同时确保对象包含在内，然后将其灰度化。接着调整图像尺寸，以生成不同尺寸的图像序列。关于图像组、主体对象/背景，以及每张图像的原始尺寸和裁剪后尺寸，详见表1 。每个组中的一个图像实例见图 1 和 2 。

图构建按照Boykov 和 Funka-Lea (2006年）中的做法，我们简要介绍如何将图像分割表述为最大流/最小割问题，以及如何编写基于边界的

[3]代码发布于 https://github.com/wang-yuyan/warmstart-graphcut-algorithms-pulic。[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

表1：图像组描述

图像组	物体、背景	原始尺寸	裁剪后尺寸
BIRDHOUSE	木制鸟舍，后院	1280, 720	600, 600
HEAD	一个人的头部、建筑物	1280, 720	600, 600
SHOE	一只鞋、地板和其他玩具	1280, 720	600, 600
DOG	伯恩山犬，草坪	1920, 1080	500, 500



(a) Birdhouse



(b) 头部



(c) 鞋子



(d) 狗

图 1： 每个组中的原始图像示例。

目标函数。输入是一幅像素集合为 V 的图像，以及两组**种子** $\mathcal{O}$， $\mathcal{B}$，它们分别是预先指定为对象或背景的像素（通常由人类专家选取），以简化分割任务。令 I_v 表示像素 v 的**强度**（或灰度）。对于任意两个像素 p， q，在对象 / 背景分割结果中将它们分开，会产生大小为 $\beta_{p,\ q}$ 的**惩罚项**。如果 p， q 是相邻像素，即 p 和 q 要么位于同一列的相邻行，要么位于同一行的相邻列，那么 $\beta_{p,\ q} = C\exp(-\frac{(I_p - I_q)^2}{2\sigma^2})$，其中 C 是一个相对较大的常数标量；否则该值为 0。因此，$\beta_{p,\ q}$ 会随着相邻 p 和 q之间对比度的增强而增大。

对于给定的解，令 J 表示对象像素。**基于边界的**目标函数是所有像素对的惩罚项之和：$\max_J \sum_{p\in J,\ q\notin J} \beta_{p,\ q}$， 对于满足 $\mathcal{O} \subseteq J$， $\mathcal{B} \subseteq V \setminus J$ 的 J。惩罚项仅施加于对象边界。最佳分割会使总惩罚项最小，



(a) Birdhouse



(b) 头部

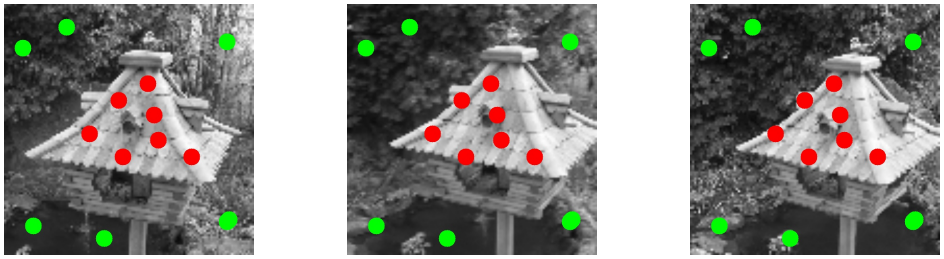


(c) 鞋子



(d) 狗

图2： 每组中的裁剪灰度图像。



（a）图像 1　　（b）图像 5　　（c）图像 10

图 3： 120×120 像素 BIRDHOUSE 序列中第一张、第五张和最后一张图像上的种子。红色表示对象，绿色表示背景。

从而在边界两侧最大化对象与背景之间的对比度，同时满足种子施加的约束。

这等价于在如下图上求解最大流/最小割问题。设节点集合为所有像素加上两个终端节点：对象终端 s（源点）和背景终端 t（汇点）。我们添加如下弧：（1）从 s 到 $\mathcal{O}$中的每个节点，容量为一个很大的值 M；（2）从 $\mathcal{B}$中的每个节点到 t，容量同样为 M；（3）对于每一对节点 p、$q \in V$（包括种子），添加两条弧（p、q）和（q、p），容量为 $\beta_{p、q}$。值 M 应确保这些弧永远不会出现在最优割中。流从 s 流向 t。对于一幅 $n \times n$ 像素的图像，该图是稀疏的，具有 $O(n^2)$ 个节点以及 $O(n^2)$ 条弧。理论链接 对于图像序列，构造的图是第 3 节中设定的推广。这些图构成二维网格，并共享相同的网络结构，唯一的差异在于容量向量。此外，第 3 节还作出了其他假设，这些假设也对应于图像的性质。1 或 M 边容量假设意味着对象和背景像素的灰度之间存在极端对比度。 d-局部假设指出，从一张图像到下一张图像，新的对象和背景像素在地理位置上彼此接近，这意味着对象的形状和位置只会发生微小变化。我们的图像序列并不严格满足这些性质。然而，在我们所有的实验中，结论对于理论假设的适度违背仍然稳健，这表明热启动在实践中可以在当前理论界限之外带来益处。

详细实验设置 每个图像序列包含 10 幅图像，且它们共享同一组种子，因此构造的图具有相同的结构。种子见图 3 ，对应于 BIRDHOUSE。从第二幅图像开始，我们复用前一幅图像的最大流解，并将流传递给算法1。在可行性投影期间，我们选取一个节点，通过寻找一条投影路径并沿该路径发送流，持续减少其超额/亏空，直到超额/亏空为 0。如第 3 节所述，我们优先选择不经过 s 和 t 的投影路径，因为这些修改会保持总体流值不变；只有在不存在其他路径时，我们才将流送回 s ，并从 t 送出。

我们分别比较 Edmonds-Karp 算法和 Dinic 算法的冷启动与热启动。对于 Edmonds-Karp 算法和 Dinic 算法的热启动，我们使用广度优先搜索（BFS）来寻找此类投影路径。对于热启动的 Dinic 算法，我们使用 BFS 子程序，而非 Dinic 算法中预期使用的子程序，因为构建层次图的开销比运行 BFS 更耗时。这是因为投影路径较短。

我们使用 $n \times n$ 像素图像作为 $n \in \{30, 60, 120\}$。数值上，$\beta_{\cdot,\cdot}$ 定义中的容量 σ 为 50，而C 为 100。为使容量为整数，所有 $\beta_{p,q}$'s 均向下取整至最接近的整数。请注意，根据定义，$\beta_{p,q} \leq C$ 。我们令 $M = C|V|^2$ ，使该项足够大。

所有实验均在配备 Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz 和 24G 内存的设备上运行。我们记录了两种算法的墙钟运行时间。许多图像处理工具和



(a) 图像 1　　(b) 图像 5　　(c) 图像 10

图 4: 120 × 120 像素 BIRDHOUSE 序列中第一幅、第五幅和最后一幅图像上的割（红色）.

函数基于图像分割GitHub 项目（Jiang，2017）。

表 2：冷启动/热启动 Ford-Fulkerson 算法和 Edmonds-Karp 算法的平均运行时间，以及热启动节省的时间百分比

图像组	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%

表3：冷启动/热启动 Ford-Fulkerson 算法的平均运行时间、热启动节省时间百分比，以及 Dinic算法

图像组	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%

结果 我们首先展示，基于边界的图像分割方法能够生成合理的割。例如，图 4 展示了来自 120 × 120 BIRDHOUSE 序列的割。其他示例请参见附录 6 。随后，我们比较冷启动和热启动 Ford-Fulkerson 算法的运行时间。由于所有算法都返回最优流，因此没有可用于衡量的解的定性方面。表 2 和 3展示了 Edmonds-Karp 算法和 Dinic算法在所有实验设置下的结果，其中行表示图像组，列表示图像尺寸。每个条目的格式为“冷启动时间（秒）/ 热启动时间（秒），热启动时间节省（%）”。

这些结果表明，在所有设置下，热启动 Ford-Fulkerson 算法都能显著提高效率。此外，冷启动和热启动的运行时间都会随图像宽度 n呈多项式增长，但热启动的增长速度更慢，因此在大规模网络上可能更具吸引力。在使用 Edmonds-Karp 算法的图像组 DOG 中，这一点最为明显：在 30×30 像素的图像上，热启动时间是冷启动时间的 60%，而在 120×120 像素的图像上仅为 20%。这些结论对 Edmonds-Karp 算法和 Dinic 算法均成立，不过在较小的数据集上，Dinic 算法的效率略高。

接下来，我们将更详细地考察冷启动/热启动的执行过程，并以 120 × 120 BIRDHOUSE Edmonds-Karp 的序列为例（表 4）。该表给出了增广

路径（“平均长度”）以及在图像序列中找到的路径平均数量（“平均数量”）。完整数据请参见附录 6。

表4：冷启动与热启动 Ford-Fulkerson 算法中投影和增广路径的比较，取自 120×120 BIRDHOUSE 图像序列的前 5 张图像

图像编号	冷启动 增广路径编号	冷启动 增广路径 平均长度	热启动 投影路径 #	热启动 投影路径 平均长度	热启动 增广路径 #	热启动 增广路径 平均长度
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

表 4 中的结果表明，投影后的可行流通常仅略微次优，这正是热启动高效的关键。前一张图像上的最大流是使用可行性投影算法进行热启动的良好起点。平均而言，将前一张图像上的最大流向下取整以满足新的边容量后，总超额/缺额为（1.75 ± 0.44）%，占实际最大流值的比例。此外，修正超额/缺额后可得到近乎最优的流。事实上，投影能快速给出可行流，恢复（96 ± 6）% 的最大流。

促成热启动效率的另一个因素是投影路径查找子程序。回想一下，冷启动和热启动都使用相同的广度优先搜索子程序，以查找一个 s、t 增广路径或投影路径。第3 节的理论表明，投影步骤中的路径查找耗时会更少。为了从实证角度展示这一点，我们收集了冷启动和热启动所找到的增广路径/可行性投影路径数量及其平均长度数据。总体而言，与冷启动相比，热启动的投影路径平均更短，这表明每条路径的广度优先搜索运行时间将大幅节省。尽管我们在表 BIRDHOUSE 4中的图像上展示了这一点，但这对其他数据集也同样成立，相关数据见附录6。即使超额/亏空较大，这也解释了算法为何仍然高效。这表明第3 节提出的理论预期能够预测实证性能。

5 结论

我们展示了如何为计算流的 Ford-Fulkerson 算法进行热启动，并证明了强有力的理论结果，同时给出了算法良好性能的实证证据。我们进一步细化了分析，以刻画利用短投影路径传输过剩流所带来的收益，并表明这些情形在图像分割应用中十分普遍。

仍有许多有趣的挑战有待解决。首先，已有许多计算流的算法，若能探索这些方法是否也能以类似方式加速，将会很有意思。一个技术障碍在于如何处理预测偏低和偏高的情况，尤其是预测导致不可行流时。更一般地说，网络流问题可以写成线性规划。另一个方向是寻找求解一般线性规划的算法，并借助精心选择的预测来改进这些算法。

参考文献

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). 网络流：理论、算法与应用. Prentice Hall。¨

Altner, D. S. and Ergun, O. (2008). 借助扩展来计算稳健最小割，快速求解最大流问题的在线序列。载于 Perron, L. and Trick, M. A.（编），约束编程中的人工智能与运筹学技术融合：组合优化问题，第五届国际会议 *CPAIOR 2008*，法国巴黎，*2008*年*5*月*20—23*日，论文集，计算机科学讲义第5015卷，第283–287页。Springer。Antoniadis, A., Gouleakis, T., Kleer, P., and Kolev, P. (2020). 带机器学习建议的秘书问题与在线匹配问题。载于 Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H.（编），神经信息处理系统进展*33*：神经信息处理系统年度会议 *2020*，*NeurIPS 2020*，*2020*年*12*月*6—12*日，线上。Balcan, M., DeBlasio, D. F., Dick, T., Kingsford, C., Sandholm, T., and Vitercik, E. (2021). 学习高性能算法需要多少数据？数据驱动算法设计的泛化保证。载于 Khuller, S. and Williams, V. V.（编），*STOC ' 21*：第*53*届 *ACM SIGACT*计算理论研讨会，线上活动，意大利，*2021*年*6*月*21—25*日，第919–932页。ACM。Bhadra, S., Kundu, A., and Khatua, S. (2020). 使用 Dinic 算法优化城市交通网络中的道路交通拥堵。载于 Abraham, A., Sasaki, H., Rios, R., Gandhi, N., Singh, U., and Ma, K.（编），仿生计算与应用创新——第*11*届仿生计算与应用创新国际会议（*IBICA 2020*）论文集，会议于*2020*年*12*月*16—18*日举行，智能系统与计算进展第1372卷，第372–379页。Springer。Boykov, Y. and Funka-Lea, G. (2006). 图割与高效的 n 维图像分割。国际计算机视觉杂志，70(2):109–131。Boykov, Y. and Kolmogorov, V. (2004). 视觉能量最小化中最小割/最大流算法的实验比较。*IEEE*模式分析与机器智能汇刊，26(9):1124–1137。

Boykov, Y. Y. and Jolly, M.-P. (2001). 用于 n 维图像中对象最优边界与区域分割的交互式图割。载于第八届 *IEEE* 国际计算机视觉会议论文集。*ICCV 2001*，第1卷，第105–112页。IEEE。Chandran, B. G. and Hochbaum, D. S. (2009). 最大流问题中伪流算法与推送-重标记算法的计算研究。运筹学，57(2):358–376。

Chen, J. Y., Silwal, S., Vakilian, A., and Zhang, F. (2022a). 借助学习预测加速基础图算法。载于 Chaudhuri, K., Jegelka, S., Song, L., Szepesv´ari, C., Niu, G., and Sabato, S.（编），国际机器学习会议，*ICML 2022*，*2022*年*7*月*17—23*日，美国马里兰州巴尔的摩，机器学习研究论文集第162卷，第3583–3602页。PMLR。Chen, L., Kyng, R., Liu, Y. P., Peng, R., Gutenberg, M. P., and Sachdeva, S. (2022b). 近线性时间内的最大流与最小费用流。$CoRR$，abs/2203.00671。Cherkassky, B. V. and Goldberg, A. V. (1997). 关于最大流问题推送—重标记方法的实现。$Algorithmica$，19(4):390–410。

Dinitz, M., Im, S., Lavastida, T., Moseley, B., and Vassilvitskii, S.（2021）。通过学习型对偶变量加速匹配。载于 Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. 编，神经信息处理系统进展第*34*卷：*2021*年神经信息处理系统年度会议，*NeurIPS 2021*，*2021*年*12*月*6*日至*14*日，线上，第10393–10406页。Dinitz, Y.（2006）。Dinitz算法：原始版本与Even版本。理论计算机科学：纪念*Shimon Even*论文集，第218–240页。Edmonds, J. and Karp, R. M.（1972）。网络流问题算法效率的理论改进。《*ACM*学报》，19（2）：248–264。Ford, L. R. and Fulkerson, D. R.（1956）。网络中的最大流。《加拿大数学杂志》，8：399–404。Gupta, R. and Roughgarden, T.（2017）。特定应用算法选择的PAC方法。《*SIAM*计算杂志》，46（3）：992–1017。Im, S., Kumar, R., Petety, A., and Purohit, M.（2022）。简约型学习增强缓存。载于 Chaudhuri, K., Jegelka, S., Song, L., Szepesv´ari, C., Niu, G., and Sabato, S. 编，国际机器学习会议，*ICML 2022*，*2022*年*7*月*17*日至*23*日，美国马里兰州巴尔的摩，机器学习研究论文集第162卷，第9588–9601页。PMLR。Im, S., Kumar, R., Qaem, M. M., and Purohit, M.（2021a）。带预测的非先知调度。载于 Agrawal, K. and Azar, Y. 编，*SPAA '21*：第*33*届 *ACM*算法与体系结构并行性研讨会，线上活动，美国，*2021*年*7*月*6*日至*8*日，第285–294页。ACM。Im, S., Moseley, B., and Pruhs, K.（2021b）。拟阵交覆盖问题。《运筹学快报》，49（1）：17–22。Jiang, J.（2017）。图像分割。https://github.com/julie-jiang/image-segmentation/。Kelner, J. A., Lee, Y. T., Orecchia, L., and Sidford, A.（2014）。一种用于无向图近似最大流的近线性时间算法及其多商品推广。载于 Chekuri, C. 编，第*25*届 *ACM-SIAM*离散算法研讨会论文集，*SODA 2014*，美国俄勒冈州波特兰，*2014*年*1*月*5*日至*7*日，第217–226页。SIAM。Kraska, T., Beutel, A., Chi, E. H., Dean, J., and Polyzotis, N.（2018）。学习型索引结构的优势。载于 Das, G., Jermaine, C. M., and Bernstein, P. A. 编，*2018*年国际数据管理会议论文集，*SIGMOD 2018*，美国得克萨斯州休斯敦，*2018*年*6*月*10*日至*15*日，第489–504页。ACM。Lattanzi, S., Lavastida, T., Moseley, B., and Vassilvitskii, S.（2020）。通过学习型权重实现在线调度。载于 Chawla, S. 编，*2020*年 *ACM-SIAM*离散算法研讨会论文集，*SODA 2020*，美国犹他州盐湖城，*2020*年*1*月*5*日至*8*日，第1859–1877页。SIAM。Lattanzi, S., Moseley, B., Vassilvitskii, S., Wang, Y., and Zhou, R.（2021）。鲁棒在线相关聚类。载于 Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. 编，神经信息处理系统进展第*34*卷：*2021*年神经信息处理系统年度会议，*NeurIPS 2021*，*2021*年*12*月*6*日至*14*日，线上，第4688–4698页。Lavastida, T., Moseley, B., Ravi, R., and Xu, C.（2021）。用于在线匹配、流和负载均衡的可学习且实例鲁棒的预测。载于 Mutzel, P., Pagh, R., and Herman, G. 编，第*29*届

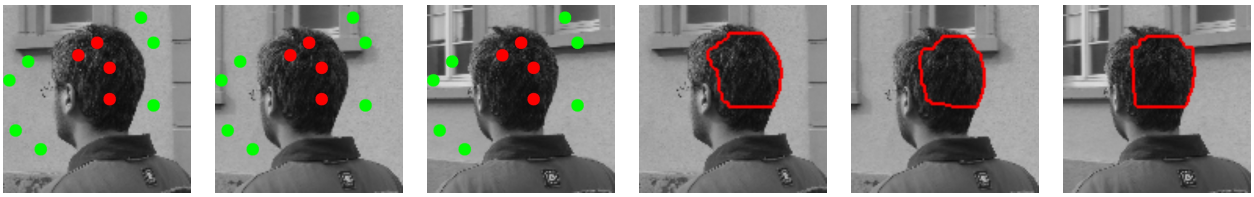
欧洲算法年度研讨会，*ESA 2021*，*2021*年*9*月*6*日至*8*日，葡萄牙里斯本（线上会议），LIPIcs第204卷，第59:1–59:17页。Schloss Dagstuhl - Leibniz信息学中心。Lindermayr, A. and Megow, N.（2022）。用于非先知调度的排列预测。载于 Agrawal, K. and Lee, I. A. 编，*SPAA '22*：第*34*届 *ACM*算法与体系结构并行性研讨会，美国宾夕法尼亚州费城，*2022*年*7*月*11*日至*14*日，第357–368页。ACM。Lu, P., Ren, X., Sun, E., and Zhang, Y.（2021）。带预测的广义排序。载于 Le, H. V. and King, V. 编，第*4*届算法简洁性研讨会，*SOSA 2021*，线上会议，*2021*年*1*月*11*日至*12*日，第111–117页。SIAM。Lykouris, T. and Vassilvitskii, S.（2021）。利用机器学习建议实现竞争性缓存。《*ACM*学报》，68（4）：24:1–24:25。Mitzenmacher, M. and Vassilvitskii, S.（2022）。带预测的算法。65（7）：33–35。Polak, A. and Zub, M.（2022）。学习增强型最大流。*CoRR*，abs/2207.12911。Purohit, M., Svitkina, Z., and Kumar, R.（2018）。利用机器学习预测改进在线算法。载于 Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. 编，神经信息处理系统进展第*31*卷：*2018*年神经信息处理系统年度会议，*NeurIPS 2018*，*2018*年*12*月*3*日至*8*日，蒙特´利尔，加拿大，第9684–9693页。Sakaue, S. and Oki, T.（2022）。基于离散凸分析的预测型算法热启动框架。载于神经信息处理系统进展第*35*卷：*2022*年神经信息处理系统年度会议，*NeurIPS 2022*，。Sherman, J.（2013）。近线性时间内求近似最大流。载于第*54*届 *IEEE* 计算机科学基础年度研讨会，*FOCS 2013*，*2013*年*10*月*26*日至*29*日，美国加利福尼亚州伯克利，第263–269页。IEEE计算机学会。Vineet, V. and Narayanan, P.（2008）。CUDA切割：在图形处理器上快速进行图切割。载于*2008*年 *IEEE*计算机学会计算机视觉与模式识别会议研讨会，第1–8页。IEEE。

6 附录

我们描述了实验设置，并提供了更多收集的数据和结果。更多内容请参见
https://github.com/wang-yuyan/warmstart-graphcut-algorithms-pulic。

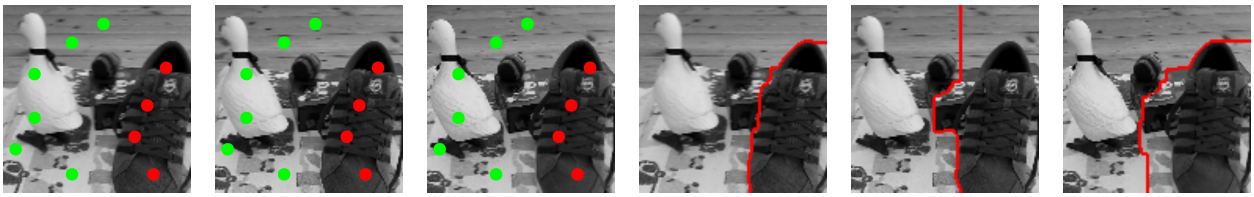
关于种子和割线的选择 回想一下，在同一序列的 10 张图像中，种子像素始终固定。这里需要指出，种子的选择（种子数量及其位置）会影响找到的最小割解。不过，只要种子能得出一个接近真实前景/背景边界的合理解，冷启动与热启动的比较结论就不会因种子变化而改变。

在第 4 节中，我们展示了 120×120 像素 BIRDHOUSE序列中第 1、5 和 10 张图像上的种子和最优割，见图 3 和 4。此外，这里还展示了其他 120×120 序列中第 1、5 和 10 张图像上的种子及其生成的割。HEAD 对应的内容见图 5， SHOE 对应的内容见图 6，而DOG 对应的内容见图 7。



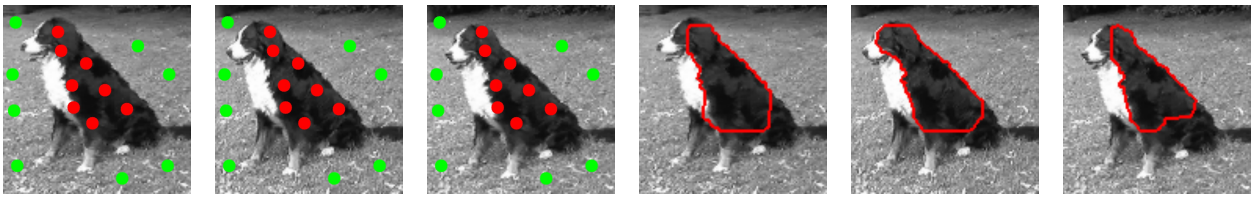
(a) 图像 1，种子 (b) 图像 5，种子 (c) 图像 10，种子 (d) 图像 1，割 (e) 图像 5，割 (f) 图像 10，割

图 5： 120×120 像素 HEAD 序列中第一幅、第五幅和最后一幅图像上的种子及其对应的割。红色种子表示对象，绿色种子表示背景，红线表示割。



(a) 图像 1，种子 (b) 图像 5，种子 (c) 图像 10，种子 (d) 图像 1，割 (e) 图像 5，割 (f) 图像 10，割

图 6：在 120×120 像素 SHOE 序列中，第一张、第五张和最后一张图像上的种子及生成的割线。红色种子表示对象，绿色种子表示背景，红线表示割线。



(a) 图像 1，种子 (b) 图像 5，种子 (c) 图像 10，种子 (d) 图像 1，割 (e) 图像 5，割 (f) 图像 10，割

图 7： 120×120 像素 DOG 序列中第一张、第五张和最后一张图像上的种子及其对应的割线。对象的红色种子，背景的绿色种子，表示割线的红线。

当我们选择一个种子时，会在目标种子周围绘制一个二维球，并将该球内的每个像素也设为种子。我们发现，这种做法比简单地选择单个像素作为

种子更有效。当我们从低分辨率（30×30）图像切换到高分辨率（120×120）图像时，会根据每边的像素数量按比例调整该球的半径。在 30×30、 60×60、 120×120 像素图像中，该球的半径分别为 1、2 和 4 个像素。换句话说，如果将不同分辨率的图像拉伸或压缩到相同大小，那么从几何角度看，该球的面积大致相同。我们还发现，与固定像素半径相比，这种方法更加有效，尽管分辨率发生了变化。



(a) 30×30 (b) 60×60 (c) 120×120

图 8：不同分辨率下，同一图像中每个种子所覆盖的面积。

请注意，尽管在整个图像序列中种子的位置保持不变，但从低分辨率图像切换到高分辨率图像时，我们仍可能需要提供更多种子。直观地说，模糊处理会减弱对象内部像素之间细微的对比度，使几何形状更容易被捕捉。 30×30 和 60×60 序列上的种子和最小割结果可在本节开头所链接的 GitHub 中上传的代码目录中找到。

关于热启动的更多奥妙 在正文中，我们从理论和实证两方面证明，热启动带来的运行时间节省主要归因于：

- 该算法能够利用较短的投影路径，将过剩流重新路由至存在亏空流的节点，从而快速将预测流投影为可行流。
- 经过可行性投影后，得到的流仅略微次优，因此热启动只需经过较少的增广路径即可达到最优流。

这里我们提供更多结果来支持这两项主张。为了展示可行性投影步骤之后的总超额/缺额（取较大者）水平以及流值，我们给出两个比率：总超额/缺额与最大流之比（表5），以及可行流值与最大流之比（表6）。可以看出，总超额/缺额通常并非微不足道。事实上，这些数值相当高；如果算法没有以恰当的方式处理超额/缺额（例如将所有超额都送至源点），流值可能会大幅降低。我们的可行性投影能够妥善决定如何利用投影路径来弥补超额/缺额，从而输出流值几乎达到最优的可行流。

表 5：热启动中总超额/缺额与最大流值的平均比率

图像组	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

为表明投影路径较短这一结论大体上适用于所有图像组，我们给出了序列中前 5 张图像上增广路径和投影路径的平均长度（“平均长度”），以及找到的路径数量（“增广路径数”和“投影路径数”）：对于 120×120 HEAD 序列，见表 7；对于 120×120 SHOE 序列，见表 8；对于 120×120 DOG 序列，见表 9。请注意， 120×120 BIRDHOUSE 序列的对应表格（表 4）位于第 4 节。

表 6： 热启动时，可行性投影后的流值与最大流值之比的平均值 rt

图像组	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

表 7：冷启动与热启动 Ford-Fulkerson 算法中投影和增广路径的比较， 120×120 HEAD 图像序列中的前 5 张图像

图像编号	冷启动 增广路径 #	冷启动 增广路径 平均长度	热启动 投影路径 #	热启动 投影路径 平均长度	热启动 增广路径 #	热启动 增广路径 平均长度
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

表 8：冷启动和热启动 Ford-Fulkerson 算法中投影与增广路径的比较，取自 120×120 SHOE 图像序列的前 5 张图像

图像编号	冷启动 增广路径编号	冷启动 增广路径 平均长度	热启动 投影路径 #	热启动 投影路径 平均长度	热启动 增广路径 #	热启动 增广路径 平均长度
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

表 9：冷启动和热启动 Ford-Fulkerson 算法中投影与增广路径的比较，取自 120×120 DOG 图像序列中的前 5 张图像

图像编号	冷启动 增广路径编号	冷启动 增广路径 平均长度	热启动 投影路径 #	热启动 投影路径 平均长度	热启动 增广路径 #	热启动 增广路径 平均长度
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

此外，我们还展示了这两种其他图像尺寸/分辨率 30×30 和60×60下表格的等价结果，分别针对图像组 HEAD （表 10 和 11）以及 SHOE （表 12 和 13）。对于这两个组，所有三种尺寸的序列具有相同的种子位置。可以看到，冷启动 Ford-Fulkerson 算法中增广路径的平均长度大致随图像宽度成比例增长。热启动可行性投影期间，投影路径的平均长度也会随图像宽度的增大而增长，但增长速度略慢于前者。这可能使热启动在高分辨率图像上更具优势。

省略的数据表和其他实验结果可在上传的程序目录中找到（说明请参阅所链接 GitHub 仓库中的 README.md 文件）。

表 10：冷启动和热启动 Ford-Fulkerson 算法中投影与增广路径的比较， 30×30 HEAD 图像序列中的前 5 张图像

图像编号	冷启动 增广路径编号	冷启动 增广路径 平均长度	热启动 投影路径 #	热启动 投影路径 平均长度	热启动 增广路径 #	热启动 增广路径 平均长度
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

表 11：冷启动与热启动 Ford-Fulkerson 算法中投影和增广路径的比较，取自 60×60 HEAD 图像序列的前 5 张图像

图像编号	冷启动 增广路径 #	冷启动 增广路径 平均长度	热启动 投影路径 #	热启动 投影路径 平均长度	热启动 增广路径编号	热启动 增广路径 平均长度
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

表 12：冷启动和热启动 Ford-Fulkerson 算法中投影与增广路径的比较，取自 30×30 SHOE 图像序列的前 5 张图像

图像编号	冷启动 增广路径 #	冷启动 增广路径 平均长度	热启动 投影路径 #	热启动 投影路径 平均长度	热启动 增广路径 #	热启动 增广路径 平均长度
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

表 13：冷启动和热启动 Ford-Fulkerson 算法中投影与增广路径的比较，来自 60×60 SHOE 图像序列的前 5 个图像

图像编号	冷启动 增广路径 #	冷启动 增广路径 平均长度	热启动 投影路径 #	热启动 投影路径 平均长度	热启动 增广路径 #	热启动 增广路径 平均长度
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