Formal Primal-Dual Algorithm Analysis

Mohammad Abdulaziz
King's College London, United Kingdom

Thomas Ammer
King's College London, United Kingdom

Abstract

We present an ongoing effort to build a framework and a library in Isabelle/HOL for formalising primal-dual arguments for the analysis of algorithms. We discuss a number of example formalisations from the theory of matching algorithms, covering classical algorithms like the Hungarian Method, widely considered the first primal-dual algorithm, and modern algorithms like the Adwords algorithm, which models the assignment of search queries to advertisers in the context of search engines.

2012 ACM Subject Classification Theory of computation → Discrete optimization; Theory of computation → Invariants; Theory of computation → Program verification

Keywords and phrases Bipartite Matching, Graph Algorithms, Isabelle/HOL, Formal Verification,

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

The Primal-dual paradigm for analysing algorithms is one of the most successful. Its history spans more than 70 years, with the Hungarian Method [27] being one of the first algorithms for solving weighted bipartite matchings to follow this paradigm. Since then, the paradigm has been used to design algorithms for problems in combinatorial optimisation, including Edmonds' algorithm for weighted matching in general graphs [16], all the way to modern analyses of online matching algorithms [13] and some of the fastest algorithms for solving flow problems [12]. In addition to exact algorithms, the primal-dual approach is also cornerstone to the design for a majority of approximation algorithms for optimisation problems, like MaxSAT, set cover, Steiner trees, etc. (cf. Part II of Vazirani's seminal book on approximation algorithms, which is dedicated to the primal-dual method).

In this work we present a series of formal analyses of primal-dual matching algorithms. Our analyses cover a range of primal-dual algorithms: from the HM, arguably the first primal-dual algorithm, to more probabilistic arguments used to analyse online algorithms, including the well-known Adwords algorithm [31] and the RANKING algorithm [25]. Our longer term goal is to create a formal library of lemmas and reasoning principles that aid in the analysis of primal-dual algorithms.

Availability. We build on an ongoing effort to build a library of combinatorial optimisation in Isabelle/HOL [1]. We plan to integrate our work in that library. In case of acceptance, we will provide a persistent link to an archived version of the formalisations we present here.

Theory Background. We assume the reader to be familiar with basic graph theory, such as vertices $\mathcal{V}$, edges $\mathcal{E}$ and paths in graphs, bipartiteness, and adjacency and incidence matrices encoding graphs. Because it eases the formalisation, we identify a graph with its edges and define $\mathcal{V} = \bigcup \mathcal{E}$. A *matching* $\mathcal{M}$ in a graph $\mathcal{E}$ is a vertex-disjoint subset of the edges in $\mathcal{E}$. Matchings covering all vertices are *perfect*. We search for matchings that are optimum w.r.t. an optimisation objective e.g. the maximisation/minimisation of accumulated real weights.

In a primal-dual (PD) method, we maintain an upper bound for the value of the solution, and finally obtain a solution whose value equals the final upper bound (or is close to it),

which implies optimality (or sufficient proximity). We encode the optimisation problem into linear (in-)equalities alongside a linear optimisation objective (*linear program/LP*), e.g. $\mathsf{max}\{c^T x.\, Ax \leq b, x \geq 0\}$ in linear algebra notation. A is usually the incidence matrix, a primal solution is a vector $x_{\mathcal{M}} \in \{0,1\}^{|\mathcal{E}|}$ encoding a matching, and a dual solution π is a $|\mathcal{V}|$-dimensional vector/function $\pi : \mathcal{V} \to \mathbb{R}$ ("potential"). We write accumulated weights as $w(E)$ for $E \subseteq \mathcal{E}$, and $\pi(\mathcal{V})$ for the potentials π.

For $\mathsf{max}\{c^T x.\, Ax \leq b, x \geq 0\}$, the *primal*, there is a *dual* $\mathsf{min}\{b^T y.\, A^T y \geq c, y \geq 0\}$. Vectors x and y satisfying these constraints are *feasible primal* and *dual solutions*, for which $c^T x \leq b^T y$ holds (*weak duality WD*). Furthermore, if $((1-\delta)A\hat{x} - b)^T \hat{y} = 0$ and $(1-\delta)(A^T\hat{y} - c)^T\hat{x} = 0$, then $(1-\delta)c^T\hat{x} = b^T\hat{y}$ and hence both $(1-\delta)\hat{x}$ and $\hat{y}$ are optimum solutions to the respective LPs (*complementary slackness CS*). If $\delta = 0$, $\hat{x}$ is an exact optimum, otherwise $\hat{x}$ satisfies an approximation guarantee. For further details, e.g. other LP types and their versions of WD and CS, we refer to the literature [35, 26].

A PD method starts with a feasible dual solution and a primal candidate solution which together satisfy CS [20] (scaled by $1-\delta$ for approximation). It iteratively changes both solutions to bring the primal closer to feasibility while maintaining CS. These steps are *primal-dual adjustments (PDA)*. When the primal solution becomes feasible, it is also optimum.

Formalisation. Our work here focuses on presenting aspects related to formal reasoning about primal-dual analyses. The main design choice regarding that is the representation of LPs: we reuse matrices [37] that others used for verifying LP theory, e.g. strong duality [33] and the simplex algorithm [36]. The mathematical core of the arguments, namely, CS and WD, is purely matrix-based and then connected to graphs by translating lemmas, which relate matchings and potentials on graphs to feasible LP solutions over matrices and vectors. The rest of the paper focuses on the reasoning of primal-dual analyses.

Other design choices peripheral to primal-dual reasoning include our definition of matching. We reuse matching-related Isabelle/HOL formalisations [8, 7, 6], which already contain many of the graph-theoretic concepts introduced above. Another decision is the approach used to model and verify algorithms, where we reuse an approach by other authors [2, 3, 4, 5, 7]. In summary, we formalise deterministic algorithms as functional programs, i.e. loops as recursive functions and program states (collections of the variables a program uses) as records. We prove *invariants* (properties of the state) for the initial state, and that they are preserved by the loop iterations, which shows them for the final state. For probabilistic algorithms, we use the existing Isabelle/HOL formalisation of the Giry monad [14] to model and reason about them. Algorithms have subprocedures that we assume and specify with Isabelle/HOL's locales [10]. These are contexts that fix mathematical entities ("locale constants") and assume their properties. Within the context, these are available for definitions and proofs and the constants can be instantiated later. This is a *stepwise refinement* [40] where a program instruction is decomposed into more detailed instructions.

2 Naive Maximum Weight Bipartite Matching

We search for matchings $\mathcal{M} \subseteq \mathcal{E}$ maximising $w(\mathcal{M})$ and derive a CS-based optimality criterion. (Maximum) integral solutions to $\mathsf{max}\{w^T x_{\mathcal{M}}.\, Ax_{\mathcal{M}} \leq 1, x_{\mathcal{M}} \geq 0\}$ encode a (max-weight) matching for $w : \mathcal{E} \to \mathbb{R}_0^+$ if A is the incidence matrix. $Ax_{\mathcal{M}} \leq 1$ and $x_{\mathcal{M}} \geq 0$ assert that $\mathcal{M}$ is a matching. The dual is $\mathsf{min}\{1^T\pi.\, A^T\pi \geq w, \pi \geq 0\}$, i.e. find π with minimum $\pi(\mathcal{V})$ s.t. $\forall\{u,v\} \in \mathcal{E}.\, \pi(u) + \pi(v) \geq w(\{u,v\})$ and $\forall v \in \mathcal{V}.\, \pi(v) \geq 0$. Such a π is a *feasible potential*, and w_π, defined as $w_\pi(\{u,v\}) = \pi(u) + \pi(v) - w(\{u,v\})$ for $\{u,v\} \in \mathcal{E}$, is the *slack*. For a

Algorithm 1 NaiveMaxWeightMatching $(\mathcal{E}, \mathcal{V} = L \cup R, w : \mathcal{E} \to \mathbb{R}_0^+)$

```
1 Initialise π(v) = max{w(e). e ∈ Δ(X)} for v ∈ L and π(v) = 0 for v ∈ R;
2 while True do
3     if ∃ matching M ⊆ E_π.{v. v ∈ V ∧ π(v) > 0} ⊆ ⋃M then return such an M;
4     else
5         find X where X ⊆ L or X ⊆ R with |X| > |Γ_π(X)| and compute
6         ε = min({w_π({u, v}). u ∈ X ∧ v ∉ Γ_π(X) ∧ w_π({u, v}) > 0} ∪ {π(v). v ∈ X});
7         for x ∈ X do π(v) ← π(v) − ε; for x ∈ Γ_π(X) do π(v) ← π(v) + ε;
```

potential π, v with $\pi(v) \neq 0$ is a *non-zero vertex*, and e with $w_\pi(e) = 0$ is *tight*. Tight edges form the *tight subgraph* $\mathcal{E}_\pi$. $\Gamma_\pi(X)$ are those vertices $v \in \mathcal{V} \setminus X$ where there is $\{u, v\} \in \mathcal{E}_\pi$ with $u \in X$. $\Delta_\pi(X)$ are the tight edges connecting X and $\Gamma_\pi(X)$. (Δ and Γ refer to the analogous notions without considering slacks.) We use this machinery to prove a sufficient condition for the weight-maximality of a matching:

▶ **Lemma 1** (Weight-Maximality). *Assume (1) $\mathcal{M}$ is a matching in a graph $\mathcal{E}$, (2) π is a feasible vertex potential, (3) all edges in $\mathcal{M}$ are tight, i.e. $\mathcal{M} \subseteq \mathcal{E}_\pi$, and (4) all vertices for which $\pi(v) \neq 0$ are matched. Then, $\mathcal{M}$ is a max-weight matching for $\mathcal{E}$.*

The Algorithm. Algorithm 1 works on $\mathcal{E}$ bipartite over L and R and $w : \mathcal{E} \to \mathbb{R}_0^+$. After initialising π to a feasible potential, it tries to find a tight matching covering all non-zero vertices. If there is such a matching, this is returned as an optimum solution. Otherwise, it performs a *primal-dual adjustment (PDA)* behind which the intuition is to move the primal (matching that matches as many non-zero vertices as possible) and the dual solution π closer together. If $\mathcal{E}$ is bipartite and there is no matching in $\mathcal{E}_\pi$ covering the non-zero vertices, we can find non-zero vertices X with $|X| > |\Gamma_\pi(X)|$, as used for the PDA. The main invariant of this algorithm is the feasibility of π (Invariant N1). We have another invariant N2 saying that the vertex potentials $\pi(v)$ are integer multiples of the same real constant α, ensuring termination for a specific class of edge weights. We need two lemmas for correctness:

▶ **Lemma 2.** *Let π be feasible for $\mathcal{E}$ and $w : \mathcal{E} \to \mathbb{R}_0^+$, and S be a set of vertices. Also, assume there is no $e \in \mathcal{E}$ with $e \subseteq S$, and $\epsilon \geq 0$, $\epsilon \leq w_\pi(e)$ for all e connecting S and $\mathcal{V} \setminus S \setminus \Gamma_\pi(S)$, and $\forall v \in S.\ \epsilon \leq \pi(v)$. Let $\pi'(v) = \pi(v) - \epsilon$ for $v \in S$, $\pi'(v) = \pi(v) + \epsilon$ for $v \in \Gamma_\pi(S)$, and $\pi'(v) = \pi(v)$ otherwise. π' is feasible, and $\pi'(\mathcal{V}) = \pi(\mathcal{V}) + (|\Gamma_\pi(S)| - |S|) \cdot \epsilon$.*

Proof Ideas. Take $\{u, v\} \in \mathcal{E}$, we know $\pi(u)+\pi(v) \geq w(\{u, v\})$, and $\pi'(u)+\pi'(v) \geq w(\{u, v\})$ follows by a case analysis. Also $\pi'(v) \geq 0$ because $\pi'(v) \leq \pi(v)$ only if $v \in S$, and then, $\pi'(v) = \pi(v) - \epsilon \geq 0$. The statement on $\pi'(\mathcal{V})$ follows from the definition of π'. ◀

▶ **Lemma 3.** *If Algorithm 1 terminates on $\mathcal{E}$ bipartite over L and R with $w : \mathcal{E} \to \mathbb{R}_0^+$, it returns a max-weight matching. If all $w(e)$ are integer multiples of $\alpha \in \mathbb{R}$, it terminates.*

Algorithm 1 is a typical primal dual algorithm: keep an upper bound (here $\pi(\mathcal{V})$) for the value of any solution (here $w(\mathcal{M})$ of any matching $\mathcal{M}$). If a solution with value equal to the upper bound cannot be found yet, we lower the upper bound carefully and continue. If $\mathcal{M}$ with $w(\mathcal{M}) = \pi(\mathcal{V})$ (inferred from CS) is found, however, $\mathcal{M}$ must have maximum weight.

Algorithm 2 HungarianMethod ($\mathcal{E}$, $\mathcal{V} = L \cup R$, $w : \mathcal{E} \to \mathbb{R}$, *initial mp-feasible* π)

```
1 if |L| ≠ |R| return infeasible; else initialise M = ∅;
2 while True do (flag, π', p) ← PathSearch(E, L, R, w, M, π);
3   if flag = dual-unbounded then return infeasible
4   else if flag = L-matched then return M else [M ← M ⊕ p; π ← π';]
```

3 The Hungarian Method

Even with termination, Algorithm 1 can have an exponential running time. However, one can combine PDAs with the search for augmenting paths (augpaths). For a matching $\mathcal{M}$, these are paths $p \subseteq \mathcal{E}$ such that the symmetric difference $\mathcal{M} \oplus p$ is a matching $\mathcal{M}'$ with $|\mathcal{M}'| > |\mathcal{M}|$. This guarantees termination in polynomial time. Although the principle of PD algorithms is simple, polynomial-time implementations and the verification thereof can be surprisingly hard. Because optimality criterion and path search are simpler for min-weight perfect matchings, we consider this problem instead.

These matchings are integral solutions to $\mathsf{min}\{w^T x_{\mathcal{M}}.\, Ax_{\mathcal{M}} = 1, x_{\mathcal{M}} \geq 0\}$, whose dual is $\mathsf{max}\{1^T\pi.\, A^T\pi \leq w\}$. Feasibility of the potential ("mp-feasibility") now means that $\pi(u) + \pi(v) \leq w(\{u,v\})$ for all $\{u,v\} \in \mathcal{E}$ *without* $\forall v \in \mathcal{V}.\ \pi(v) \geq 0$. WD is here that $1^T\pi \leq w^T x_{\mathcal{M}}$ for an mp-feasible π and perfect matching $\mathcal{M}$. CS is simplified to feasible $\hat{x}$ and $\hat{\pi}$ being optimum solutions for the respective LPs iff $(A^T\hat{\pi} - w)^T\hat{x} = 0$. Similarly to Lemma 4, we formalised this optimality criterion:

▶ **Lemma 4.** *If (1) $\mathcal{M}$ is a perfect matching in $\mathcal{E}$, (2) π is an mp-feasible vertex potential and (3) edges in $\mathcal{M}$ are tight w.r.t. π, then $\mathcal{M}$ is a min-weight perfect matching for $\mathcal{E}$.*

There is verified executable code for max-weight bipartite matching because we reduce this and 4 other problems to min-weight perfect matching. We extend the graph $\mathcal{E}$ to a complete bipartite graph $\mathcal{E}'$ with sides of equal size ($|L'| = |R'|$ and $\mathcal{E}' = \{\{u,v\}.\ u \in L' \wedge v \in R'\}$), ensuring the existence of a perfect matching. If edges in $\mathcal{E}' \setminus \mathcal{E}$ should be avoided, we impose a penalty weight on them. From a min-weight perfect matching for the extended weights in the new graph, we select edges forming an optimum solution for the original problem.

Top Loop. The Hungarian Method (HM) [23, 27, 28] (Algorithm 2) expects a bipartite graph given by $\mathcal{E}$, L and R, a weight function w and an initially feasible potential π. After checking for obvious infeasibility if $|L| \neq |R|$ and initialising $\mathcal{M}$ as $\emptyset$, the main loop repeatedly applies PathSearch() which returns a status flag. If *flag* says that the dual LP is unbounded, infeasibility follows. If L is matched, the current matching is returned as min-weight perfect matching for $\mathcal{E}$. Otherwise, PathSearch also returns a new feasible potential π' and an $\mathcal{M}$-augpath p. p is used to augment $\mathcal{M}$ and the algorithm continues with the next iteration.

Algorithm 2 maintains as invariants that (HM1) $\mathcal{M}$ is a matching in $\mathcal{E}$, (HM2) $\mathcal{M} \subseteq \mathcal{E}_\pi$, and (HM3) π is mp-feasible w.r.t. w and $\mathcal{E}$. Provided that $\mathcal{M}$ is a matching in $\mathcal{E}_\pi$ and π is mp-feasible, we assume three properties for $(\mathit{flag}, \pi', p) = \text{PathSearch}(\mathcal{E}, L, R, w, \mathcal{M}, \pi)$: (P1) If *flag* = *dual-unbounded* there is an mp-feasible π' with $\pi'(\mathcal{V}) > B$ for any $B \in \mathbb{R}$. (P2) If *flag* = *L-matched*, $L \subseteq \bigcup \mathcal{M}$. (P3) If *flag* = *next-iteration*, π' is mp-feasible, $\mathcal{M} \subseteq \mathcal{E}_{\pi'}$, $p \subseteq \mathcal{E}_{\pi'}$ and p is an $\mathcal{M}$-augpath. Assuming P1-P3, the following correctness lemma holds:

▶ **Lemma 5.** *Let $\mathcal{E}$ be bipartite over L and R, and π be mp-feasible w.r.t. $\mathcal{E}$ and $w : \mathcal{E} \to \mathbb{R}$. On this input, the HM terminates, and returns a min-weight perfect matching if there is one.*

Algorithm 3 *RANKING*($\mathcal{E}$ bipartite over L and R, arrival order ω for R)

$\sigma \leftarrow$ a random permutation of L; $\mathcal{M} \leftarrow \emptyset$;
for *every arriving vertex* u *in* ω **do**
  if $\exists v \in \gamma(u) \setminus \bigcup \mathcal{M}$ **then** $\mathcal{M} \leftarrow \mathcal{M} \cup \{\{\mathsf{argmin}_{v \in \gamma(u) \setminus \bigcup \mathcal{M}} \sigma(v), u\}\}$;
return $\mathcal{M}$;

Path Search. We search for augpaths $p \subseteq \mathcal{E}_\pi$ that are tight w.r.t. an mp-feasible π, for which *alternating forests (AF)* are the central tool [17]. Other authors formalised a non-executable version for the blossom algorithm [8, 7], and we provide an executable one that works well with forest growth and PDAs happening at the same time. An ϵ is determined similar to Algorithm 1 and preservation of mp-feasibility is similar to Lemma 2. Efficient search requires "caching" of slacks and a priority queue. We used *priority search trees* [29] for the queue and red-black trees for other data structures, resulting in a verified $\mathcal{O}(n \cdot (n+m) \cdot \log n)$ implementation of the HM, which is the best possible running time of a purely functional implementation. The verification effort for PathSearch was high with 8 involved invariants.

4 Online Matching

Online bipartite matching is a variant of bipartite matching where vertices in one party of the graph arrive online, one-by-one, each along with its incident edges. After a vertex' arrival, the algorithm irrevocably decides on whether any of its incident edges is included in the matching. This setting was introduced by Karp et al. [25], where they considered unweighted bipartite matching. The online model of matching has recently gained substantial interest [32, 24, 21, 22] due to the proliferation of the internet and different applications which could be modelled and understood as online matching problems. Most notable among those is the Adwords [31] algorithm, which is an idealised model of how advertisers bid on keywords to show their ads to users who search for those keywords.

RANKING [25], Adwords [31], and most other online matching algorithms have proofs that are quite complex to understand, let alone formalise. Often, this is primarily due to the complex combinatorial arguments used to show them correct. For instance, the correctness of RANKING's initial analysis by Karp et al., which was formalised earlier by Abdulaziz and Madlener [6], was improved over 6 times by a number of authors [19, 11, 13, 15, 39, 32], similarly to improvements to the analysis [13, 38] of Adwords. In this work, as part of our framework, we formalise a PD-analysis that has substantially simplified the competitiveness analysis of a number of online matching algorithms. We briefly describe those formalisations.

We first discuss the RANKING algorithm briefly, following the previous formalisation [6]. In its original form, RANKING operates as sketched in Algorithm 3: it takes as an input a bipartite graph with edges $\mathcal{E}$, and a list ω with the order of arrival of the online vertices R. The offline party L is permuted uniformly randomly, leading to a list σ. We then go through the online vertices in the order given by ω, adding to the matching an edge that, if it exists, connects the incoming vertex u to the unmatched offline vertex v with the highest ranking w.r.t. σ. The neighbourhood, i.e. all vertices a vertex u is connected to, is $\gamma(u)$. The main theorem to prove about RANKING pertains to its competitive ratio:

▶ **Theorem 1.** *The competitive ratio of RANKING for an instance with a maximum cardinality matching of size n is at least $1 - \frac{1}{e}$, i.e. $1 - \frac{1}{e} \leq \frac{\mathbb{E}_{R \sim RANKING(G,\pi)}[R]}{n}$.*

The main idea of the primal-dual analysis of RANKING is as follows: we have a primal LP

relaxation of bipartite matching $\mathsf{max}\{x_{\mathcal{M}}.\, Ax_{\mathcal{M}} \leq 1, x_{\mathcal{M}} \geq 0\}$ and its dual $\mathsf{min}\{1^T\pi.\, A^T\pi \geq 1, \pi \geq 0\}$, i.e. find π with minimum $\pi(\mathcal{V})$ s.t. $\forall u \in L, v \in R, \{u,v\} \in \mathcal{E}.\ \pi(u) + \pi(v) \geq 1$ and $\forall v \in \mathcal{V}.\ \pi(v) \geq 0$. Now, as mentioned earlier, a primal-dual analysis of an optimisation algorithm usually proceeds by proving a relationship between the primal objective and the dual objective. Proving this relation for RANKING is challenging as we need to tackle the difficulty with randomisation in RANKING, which makes it a non-standard primal-dual analysis. We can only reason about the expectation of the algorithm's output and thus we can only reason about the relationship between expected values of the primal and dual LPs.

Both LPs have to be related to or derived from the output, as in case of the two offline algorithms discussed earlier. The primal LP still corresponds to the computed matching $\mathcal{M}$, but now the main challenge is connecting the dual LP to $\mathcal{M}$: a different approach specific to the online setting is needed. In particular, the potential has to include information about the ordering of the offline vertices after they are permuted. To reason about the dual LP's expected objective, we replace the permutation step with choosing a real-numbered priority $Y_i \in [0,1]$ for each offline vertex i. This is to allow for the expectation to be over an integrable function. Then each constraint in the dual LP is instantiated as follows: $\pi(u) = g(Y_i)/F$ and $\pi(v) = (1 - g(Y_i))/F$ for $g : [0,1] \to [0,1]$ that is monotone and $0 < F \leq 1$, for every $u \in L$, $v \in R$, and $\{u,v\} \in \mathcal{E}$. This allows us to formalise the following reasoning:

$$
\begin{aligned}
\mathbb{E}[x_{\mathcal{M}}] &= \mathbb{E}[1^T \cdot (\frac{1}{F} \cdot \pi)] \quad \text{(by construction of the primal and dual LPs)} \\
&= (1^T \cdot \mathbb{E}[\pi])/F \quad \text{(linearity of expectation)} \\
&\geq 1 \cdot x^* \cdot F \quad \text{(by weak duality, where } x^* \text{ is an optimal primal solution)} \\
&= 1 \cdot |\mathcal{M}^*| \cdot F \quad \text{(by definition of the primal LP)}
\end{aligned}
$$

for any max-card matching $\mathcal{M}^*$. The last step is choosing g and F, where we set $F = 1 - 1/e$, thus proving the bound.[1]

This approach has two advantages: it is much simpler than the combinatorial approach as formalised by Abdulaziz and Madlener as the primal-dual-based proof is less than one half (3K lines) of the combinatorial one. Moreover, it provides a general framework for reasoning about online matching algorithms. We used our development to (a) formalise the competitive analysis for the vertex-weighted variant of online matching [9], where the primal and dual LPs are adapted to include the weights with the entire analysis remaining unchanged. (b) With some changes, we did a primal-dual analysis of the Adwords algorithm [31], which solves a variant of online b-matching, that models the assignment of keywords to advertisers in the context of search engines. Interested readers should refer to the formalisations.

5 Discussion

There is a rich literature on the formal analysis of algorithms, including approximation algorithms [18], matching algorithms [7], flow algorithms [3, 30]. Our work here, however, addresses a large gap in the literature, namely, the formalisation of primal-dual analyses of algorithms. We primarily focus on matching algorithms here, covering classical and modern results. Our formalisations, which are around 14K lines, cover a large variety of reasoning styles: the Hungarian Method, which is an executable practical algorithm, and variants of

[1] The expectations are taken over the distribution of priorities Y. A step we gloss over here is the equivalence of expectations over permutations and priorities.

online matching algorithms, which are primarily used to theoretically analyse online markets rather than to solve practical problems.

PD-based reasoning about the correctness of algorithms is mostly algebraic and thus leads to simpler, shorter and more textbook-style proofs. The formalised combinatorial arguments for the correctness of RANKING [6] and Berge's Lemma for the blossom algorithm [7] are of much higher complexity, come with extensive case analyses and are harder to understand than the PD-arguments for the HM and RANKING, for example. We believe PD-based reasoning can also simplify correctness arguments for minimum cost flows, for which there was only a combinatorial proof formalised so far [3]. According to standard literature [26, 34], maintenance of dual variables even leads to faster algorithms for this problem.

However, a big missing part in our work that we aim to contribute to a library [1] is the analysis of primal-dual approximation algorithms: these indeed form the majority of applications of the primal-dual paradigm in theoretical computer science. Our immediate plans are to formalise algorithms for approximating MaxSAT, set cover, load balancing, and Steiner trees, all of which are milestones in the theory of approximation algorithms.

References

1 Isabelle graph library. URL: `https://github.com/mabdula/isabelle-graph-library`.

2 Mohammad Abdulaziz. *Graph Algorithms*, page 273–298. Association for Computing Machinery, New York, NY, USA, 1 edition, 2025. URL: `https://doi.org/10.1145/3731369.3731394`.

3 Mohammad Abdulaziz and Thomas Ammer. A Formal Analysis of Capacity Scaling Algorithms for Minimum Cost Flows. In Yves Bertot, Temur Kutsia, and Michael Norrish, editors, *15th International Conference on Interactive Theorem Proving (ITP 2024)*, volume 309 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:19, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITP.2024.3`, `doi:10.4230/LIPIcs.ITP.2024.3`.

4 Mohammad Abdulaziz and Thomas Ammer. A Formal Analysis of Capacity Scaling Algorithms for Minimum-Cost Flows, 2026. `arXiv:2602.03701`, `doi:10.48550/arXiv.2602.03701`.

5 Mohammad Abdulaziz, Thomas Ammer, Shriya Meenakshisundaram, and Adem Rimpapa. A Formal Analysis of Algorithms for Matroids and Greedoids. In Yannick Forster and Chantal Keller, editors, *16th International Conference on Interactive Theorem Proving (ITP 2025)*, volume 352 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:19, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITP.2025.2`, `doi:10.4230/LIPIcs.ITP.2025.2`.

6 Mohammad Abdulaziz and Christoph Madlener. A Formal Analysis of RANKING. In *The 14th Conference on Interactive Theorem Proving (ITP)*, 2023. `doi:10.48550/arXiv.2302.13747`.

7 Mohammad Abdulaziz and Kurt Mehlhorn. A Formal Correctness Proof of Edmonds' Blossom Shrinking Algorithm, 2025. URL: `https://arxiv.org/abs/2412.20878`, `arXiv:2412.20878`.

8 Mohammad Abdulaziz, Kurt Mehlhorn, and Tobias Nipkow. Trustworthy Graph Algorithms. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.MFCS.2019.1`, `doi:10.4230/LIPIcs.MFCS.2019.1`.

9 Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online Vertex-Weighted Bipartite Matching and Single-bid Budgeted Allocations. In *The 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011. `doi:10.1137/1.9781611973082.95`.

10 Clemens Ballarin. Interpretation of Locales in Isabelle: Theories and Proof Contexts". In Jonathan M. Borwein and William M. Farmer, editors, *Mathematical Knowledge Management*, pages 31–43, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

11 Benjamin E. Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *SIGACT News*, 2008. doi:10.1145/1360443.1360462.

12 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum Flow and Minimum-Cost Flow in Almost-Linear Time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, 2022. doi:10.1109/FOCS54457.2022.00064.

13 Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized Primal-Dual Analysis of RANKING for Online Bipartite Matching. In *The 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013. doi:10.1137/1.9781611973105.7.

14 Manuel Eberl, Johannes Hölzl, and Tobias Nipkow. A Verified Compiler for Probability Density Functions. In *Programming Languages and Systems - 24th European Symposium on Programming, ESOP 2015, London, UK, April 11-18, 2015. Proceedings*, 2015. doi:10.1007/978-3-662-46669-8_4.

15 Alon Eden, Michal Feldman, Amos Fiat, and Kineret Segal. An Economics-Based Analysis of RANKING for Online Bipartite Matching. In *Symposium on Simplicity in Algorithms (SOSA)*, 2021. doi:10.1137/1.9781611976496.12.

16 Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, 1965. doi:10.6028/jres.069B.013.

17 Jack Edmonds. Paths, Trees, and Flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi:10.4153/CJM-1965-045-4.

18 Robin E*xs*mann, Tobias Nipkow, Simon Robillard, and Ujkan Sulejmani. Verified Approximation Algorithms. *Log. Methods Comput. Sci.*, 2022. doi:10.46298/LMCS-18(1:36)2022.

19 Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to Adwords. In *The 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347189.

20 Michel X. Goemans and David P. Williamson. *The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems*, page 144–191. PWS Publishing Co., USA, 1996.

21 Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. Fully Online Matching. *J. ACM*, 2020. doi:10.1145/3390890.

22 Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. Fully Online Matching II: Beating Ranking and Water-filling. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, 2020. doi:10.1109/FOCS46700.2020.00130.

23 Carl Gustav Jacobi. De investigando Ordine Systematis Aequationum Differentialum vulgarium cuiuscumque. *Journal für die reine und angewandte Mathematik*, 64:297–320, 1890. URL: https://www.lix.polytechnique.fr/~ollivier/JACOBI/Jacobi_De_Investigando.pdf.

24 Bala Kalyanasundaram and Kirk Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 2000. doi:10.1016/S0304-3975(99)00140-1.

25 R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *The 22nd ACM Symposium on Theory of Computing (STOC)*, Baltimore, Maryland, United States, 1990. doi:10.1145/100216.100262.

26 Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. doi:10.1007/978-3-642-24488-9.

27 Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, March 1955. doi:10.1002/nav.3800020109.

28 Harold W. Kuhn. Variants of the Hungarian Method for Assignment Problems. *Naval Research Logistics Quarterly*, 3:253–258, December 1956. doi:10.1002/nav.3800030404.

29 Peter Lammich and Tobias Nipkow. Proof Pearl: Purely Functional, Simple and Efficient Priority Search Trees and Applications to Prim and Dijkstra. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving (ITP 2019)*, volume 141 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:18, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITP.2019.23`, `doi:10.4230/LIPIcs.ITP.2019.23`.

30 Peter Lammich and S. Reza Sefidgar. Formalizing Network Flow Algorithms: A Refinement Approach in Isabelle/HOL. *J. Autom. Reason.*, 2019. `doi:10.1007/s10817-017-9442-4`.

31 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. AdWords and generalized online matching. *J. ACM*, 2007. `doi:10.1145/1284320.1284321`.

32 Milena Mihail and Thorben Tröbst. Online Matching with High Probability, 2021. URL: `http://arxiv.org/abs/2112.07228`, `arXiv:2112.07228`.

33 Julian Parsert. Linear Programming in Isabelle/HOL, 2024. URL: `https://arxiv.org/abs/2403.19639`, `arXiv:2403.19639`.

34 A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin ; New York, 2003.

35 Alexander Schrijver. Theory of Linear and Integer Programming. In *Wiley-Interscience Series in Discrete Mathematics and Optimization*, 1986. URL: `https://api.semanticscholar.org/CorpusID:29180149`.

36 Mirko Spasić and Filip Marić. Formalization of Incremental Simplex Algorithm by Stepwise Refinement. In Dimitra Giannakopoulou and Dominique Méry, editors, *FM 2012: Formal Methods*, pages 434–449, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

37 René Thiemann and Akihisa Yamada. Matrices, Jordan Normal Forms, and Spectral Radius Theory. *Archive of Formal Proofs*, August 2015. `https://isa-afp.org/entries/Jordan_Normal_Form.html`, Formal proof development.

38 Vijay V. Vazirani. Online Bipartite Matching and Adwords, 2022. URL: `http://arxiv.org/abs/2107.10777`, `arXiv:2107.10777`.

39 Vijay V. Vazirani. Online Bipartite Matching and Adwords (Invited Talk). In *The 47th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2022. `doi:10.4230/LIPIcs.MFCS.2022.5`.

40 Niklaus Wirth. Program Development by Stepwise Refinement. *Commun. ACM*, 14(4):221–227, April 1971. `doi:10.1145/362575.362577`.