Greediness is not always a vice: Efficient Discovery Algorithms for Assignment Problems

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 223 (2023) 43–52

Procedia Computer Science

www.elsevier.com/locate/procedia

XII Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS 2023)

Greediness is not always a vice: Efficient Discovery Algorithms for Assignment Problems

Romaric Duvignau[a,*], Ralf Klasing[b]

[a] *Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden*
[b] *CNRS, LaBRI, Université de Bordeaux, Talence, France*

Abstract

Finding a maximum-weight matching is a classical and well-studied problem in computer science, solvable in cubic time in general graphs. We introduce and consider in this work the "discovery" variant of the bipartite matching problem (or assignment problem) where edge weights are not provided as input but must be *queried*, requiring additional and costly computations. Hence, discovery algorithms are developed aiming to minimize the number of queried weights while providing guarantees on the computed solution. We show in this work the hardness of the underlying problem in general while providing several efficient algorithms that can make use of natural assumptions about the order in which the nodes are processed by the greedy algorithms. Our motivations for exploring this problem stem from finding practical solutions to maximum-weight matching in hypergraphs, a problem recently emerging in the formation of peer-to-peer energy sharing communities.

Keywords: discovery algorithms; query complexity; assignment problem; greedy algorithms

1. Introduction

As one of the probably most studied problems in computer science and discrete maths, the *assignment problem* has a very simple formulation, yet a plethora of solutions exist for its many variants and possible additional assumptions or optimization aims. Describing the problem in the same terms and conventions we use in the rest of the paper, the assignment problem consists in pairing together the member of a first set P, often referred as *Producers* or *agents* in the literature, with members of a second and disjoint set C, often referred as *Consumers* or *tasks*. The rule assumed here is of a *One-to-One* correspondence, i.e. each producer may handle at most a single consumer and vice-versa, and as not all producers may be able to serve any particular consumer, some pairs are considered non valid (a variant would consider setting a weight of 0 for those pairs but zero weights are ruled out in our formulation). For each possible pair $(p, c) \in E$ with $E \subseteq P \times C$, (p, c) is associated with a positive *weight* $w(p, c)$ that represents how much *gain* one can

* Corresponding author.
E-mail addresses: duvignau@chalmers.se (Romaric Duvignau)., ralf.klasing@labri.fr (Ralf Klasing).

1877-0509
10.1016/j.procs.2023.08.212

obtain if producer $p \in P$ is paired with consumer $c \in C$. The problem consists then in finding an *assignment* M of the consumers to the producers in order to maximize the total gain $w(M) = \sum_{(p,c)\in M} w(p,c)$. This is a well-studied problem where the Hungarian algorithm [1] computes the optimal solution in time $O(n \cdot m + n^2 \log n)$ for $n = |P|$ and $m = |E|$; see among others [2] for unbalanced assignment problems and [3] for linear-time bounded-approximation algorithms. One may observe that the problem can be alternatively formulated as finding a maximum-weight matching in the bipartite graph $G = (P \cup C, E)$, with the two formulations being equivalent and used interchangeably for convenience.

A "discovery" problem is any optimization problem where the information that is the basis of the optimization is not provided as input but must rather be *discovered* during the algorithm's execution. We extend this notion of discovery algorithms, introduced among others in [4, 5], to assignment problems. We shall study in this work the *Maximum-Weight Matching Discovery (MWMD) problem* that consists in finding a Maximum-Weight Matching (MWM) using weights that can only be obtained through explicit calls to a computationally-expensive weight function. We denote by *query complexity* the number of inspected weights used by a given algorithm to produce its solution. Since one can easily show that in general, finding the MWM requires the computation of all possible weights, we aim to investigate in this paper if approximation algorithms can reach a bounded approximation ratio while requiring the calculation of only a subquadratic number of weights in n. Our methods apply to the assignment problems (i.e., bipartite graph matchings) and can be further extended to solve a bipartite version of the hypergraph matching problem.

Contributions. Recall the greedy matching considers the edges one-by-one in decreasing order of weights and at each step, the procedure adds the current edge if both its endpoints are still available. It is a folklore result that the greedy matching algorithm M_g produces a 2-approximate matching compared with the optimal algorithm, i.e., we have $w(M_{opt}) \leq 2 \cdot w(M_g)$ where M_{opt} is the MWM on the input calculated for instance using the Hungarian algorithm; note that both the greedy and optimal matching require to inspect the value of all the weights of the input to compute their solution. The argument for the bounded approximation bound relies on two elements: the order in which the greedy algorithm considers the edges (from largest to smallest weights) and the fact that for each edge e of M_g, if e is not present in M_{opt} then it may only be "replaced" by two other edges in M_{opt}, from which one deduces the approximation bound. Our main contribution is to propose a generalization of the above argument to edge sets that are only *partially ordered*, hence allowing to deduce approximation bounds using problem-dependent *heuristic orders* on the vertex sets. In this work, we introduce in Section 2.2 "oracles" that are capable to order nodes in specific orders concerning the weights of the edges in their neighborhood without requiring any computation of the edge weights. This ordering allows us to design efficient greedy algorithms with bounded approximation ratio and requiring to compute only $O(n)$ weights when the vertices of each set are processed in a heuristic order, while still allowing some of the nodes to be processed out of their real order. Our solutions extend to hypergraph matching as well, hence making them ideal candidates to solve the problem in practical instances. We summarize our results in Table 1. "Classic-Greedy" refers there to the classical greedy algorithm ordering all edges by weight while the other algorithms are the ones developed and analysed in this work. Parameters $\beta, \gamma, \gamma_\ell, \beta_\ell$ control the quality of the heuristic orders for processing of the nodes of P and C, are respectively specified in Assumption 1, 2, 3 and 4, and are assumed to be greater than or equal to 1.

Table 1. Approximation ratios shown in this work for the one-to-one assignment discovery problem.

Algorithm	Hungarian	Classic-Greedy	Greedy-Local	ℓ-Greedy-Local	Naive-Local
Query Complexity	$m = \lvert E\rvert$		$\leq m$	$\leq (\ell+1)\cdot n$	0
No weight assumption	1	2	unbounded		
Assumption 1			$1+\beta$	unbounded	
Assumption 2			$1+\gamma$	unbounded	
Ass. 1 + Ass. 2			$1+\min\{\beta,\gamma\}$	$\beta+\gamma$	
Ass. 1 + Ass. 3			$\beta+\gamma_\ell$		unbounded
Ass. 2 + Ass. 4			$\gamma+\beta_\ell$		unbounded
Ass. 3 + Ass. 4			unbounded		

Motivations. In the context of peer-to-peer energy sharing [6], the Geographical Peer Matching (GPM) problem is introduced in [7] to efficiently compute a matching of the peers targeting the maximization of a global objective. It relies on both geographical information about the peers as well as their local matching preferences, and seeks to build an efficient bipartite hypergraph matching of the peers as advocated by the application. The matching problem involved, a variant of hypergraph matching with all hyperedges of size up to k, is not approximable within a factor of $o(k/\log k)$ in polynomial time, unless P = NP [7]. The best approximation algorithms [8, 9, 10] for solving it reach an approximation ratio slightly better than $(k+1)/2$, but they require the computation of $O(n^k)$ weights, already prohibitive in practical instances. Using the extended version of the discovery algorithms presented and analysed in this work, we obtain algorithms running in $O(kn)$ time, requiring the knowledge of $O(kn)$ weights, and achieving an approximation bound of $1+\varepsilon$ where ε is a constant controlling the precision of the heuristic order used to compute the matching. Hence, under certain assumptions occurring in practice, greedy matchings do produce bounded approximations.

Related work. Discovery algorithms have been studied in the literature for various problems on weighted graphs. However, as far as we are aware, they have not been investigated so far for the maximum-weight matching problem. Note that the time complexity of producing an optimal solution to the discovery problem is always at least as large as the one for the usual problem with all weights provided at the beginning.

Szepesvari [4] introduced the *Shortest Path Discovery Problem* (SPDP), in which the task is to discover in a given edge-weighted graph a shortest path for fixed source and target nodes. An algorithm is proposed that is shown to use a small number of queries. Experimental results on real-world instances are also presented. Caro et al. [5] generalize the SPDP to the *Optimal Path Discovery Problem*. First, they consider a broader class of cost functions, and relax the constraint that an optimal path has to be discovered, allowing the discovered path to be an α-approximation. Second, whereas in [4] the performance of algorithms was measured with the number of queries, Caro et al. [5] propose a more fine-grained performance measure, called the *query ratio*, i.e., the ratio between the number of queried edges and the least number of edge values required to solve the problem. They prove a $1 + 4/n - 8/n^2$ lower bound on the query ratio and present an algorithm whose query ratio, when it finds the optimal path, is upper bounded by $2 - 1/(n-1)$, where $n = |V|$. Finally, they implement different algorithms and evaluate their query ratio experimentally.

In [11], the authors consider the minimum spanning tree problem with *queryable uncertainty*. This concept refers to settings where the input of a problem is initially not known precisely, but exact information about the input can be obtained at a cost using queries. An algorithm with query ratio 2 is proposed in [11] for the minimum spanning tree problem, and it is shown that this query ratio is the best possible among deterministic algorithms. In [12], the authors extend the framework to cheapest set problems with queryable uncertainty that englobe previously studied problems such as the minimum spanning tree, or the minimum matroid base problem under queryable uncertainty. For the cheapest set problems with queryable uncertainty, the authors present an algorithm that makes $d \cdot \mathrm{OPT} + d$ queries, where OPT is the optimal number of queries required to solve the problem and d is the maximum cardinality of a feasible set in a given instance. An algorithm with query ratio 2 for the minimum matroid base problem is also provided in [12].

Another similar line of work considers the robust spanning tree problem with interval data. For a given graph with weight intervals specified for its edges, the goal is to compute e.g. a spanning tree that minimizes the worst-case deviation from the minimum spanning tree (also called the *regret*), over all realizations of the edge weights. This is an off-line problem, and no query operations are involved. The problem is proved NP-hard in [13]. A 2-approximation algorithm is given in [14]. Further work has considered heuristics or exact algorithms for the problem, see e.g. [15].

Plan. In Section 2, we show the hardness of the assignment discovery problem and hence the need for order oracles in order to analyse the performance of greedy algorithms for solving the discovery problem. In Section 3, we present several greedy algorithms producing a matching without querying the totality of weights, and analyse them relying on different assumptions about the order in which the nodes are processed in regard to the weights of the edges. In Section 4, we present how our algorithms extend to the one-to-many assignment problems, before concluding our work.

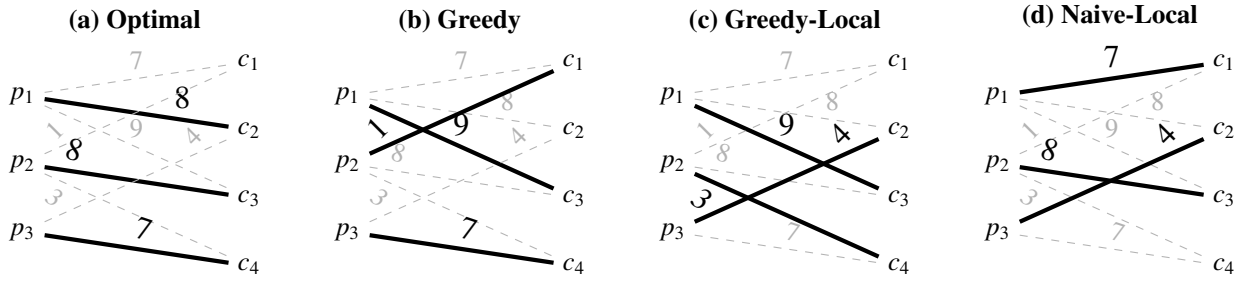

Fig. 1. Examples of matchings: (a) Optimal with weight 23, (b) Greedy with weight 17, (c) Greedy-Local with weight 16, (d) Naive-Local with weight 19. Here, $\beta = 7/3$, $\gamma = 8$, $\beta_1 = 1$, $\gamma_1 = 3$ and $\gamma_2 = 1$ the 1-Greedy-Local algorithm outputs the matching (a).

2. Order Oracles for the Assignment Discovery Problem

2.1. Preliminaries

We adopt the following conventions for the notation used hereafter. Let $G = (P \cup C, E)$ denote a bipartite graph serving as our input instance; P, a set of "agents" to match with "tasks" with $n = |P|$ the number of considered agents; C, a set of tasks with $q = |C|$; $E \subseteq P \times C$ the set of possible edges with $(p, c) \in E$ if the task c can be assigned to the agent p and $m = |E|$; $w(e)$ for $e \in E$ is the weight of edge e. We slightly abuse the w notation so as to write $w(p, c)$ to denote as well the weight of the edge (p, c) and $w(M)$ for the weight of the matching $M \subseteq E$, i.e., $w(M) = \sum_{e \in M} w(e)$. An isolated edge is any edge without any adjacent edges in G, i.e., $e = (p, c)$ is isolated if $\neg(\exists (p, c') \in C, c' \neq c \vee \exists (p', c) \in E, p' \neq p)$. In the following, we assume that all weights are strictly positive as edges with negative or zero weight are assumed to be removed from the considered input graph. The *query complexity* of an algorithm $\mathcal{A}$ is the number of weights $\mathcal{A}$ inspects in the *worst-case* in order to compute its output. Example of matchings are provided in Fig. 1 with the discovery algorithms computing them being defined in Section 3.

2.2. The need for order oracles

We first show that, without additional assumptions, any algorithm requires the computation of all possible weights in G (discarding isolated edges) in order to reach a bounded approximation ratio. Note if G has isolated edges, all such edges can be added safely without computing their weights and we can assume that the considered algorithms rather start with G', the graph obtained by stripping all isolated edges from G.

Proposition 1. *If no additional assumptions on the weights are provided, then there does not exist a bounded-approximation algorithm that computes strictly less than $m' = |E'|$ weights, with E' being the set of non-isolated edges in the input graph G.*

Proof. Suppose there exists an α-approximate algorithm $\mathcal{A}$ that always avoids the computation of at least one weight $w(p, c) > 0$ for a non-isolated edge (p, c) in a given input graph $G_w = (V, E)$ with weight function w. Since G_w has no isolated edges, there exists an edge $e' \in E$ of the form $e' = (p', c)$ with $p' \neq p$, or of the form $e' = (p, c')$ with $c' \neq c$.

Denote $M = \mathcal{A}(G_w)$ the matching obtained by running $\mathcal{A}$ on G_w. Hereafter, we will further define the weighted graph G_w by setting differently the value of $w(p, c)$ that is, by assumption, never discovered by the algorithm $\mathcal{A}$. Intuitively, the proof relies on the fact that if (p, c) is not selected then $w(p, c)$ can be arbitrarily large and if (p, c) is selected, it can be very small. Let $w(e') = 2\alpha g$ with $g = \sum_{e \in E, e \notin \{(p,c), e'\}} w(e)$. Suppose $(p, c) \notin M$. Setting $w(p, c) = 2\alpha \sum_{e \in E, e \neq (p,c)} w(e) = 2\alpha(g + 2\alpha g)$ entails that $w(M) \leq \sum_{e \in E, e \neq (p,c)} w(e) \leq \frac{w_{\text{opt}}}{2\alpha}$ as the weight of the optimal matching w_{opt} is always larger than the weight of any edge in the input graph, and in particular of the edge (p, c). Hence $\mathcal{A}$ is not α-approximate in that case. Suppose now $(p, c) \in M$. Let now $w(p, c) = \varepsilon$ with $0 < \varepsilon < g$. We have $w_{\text{opt}} \geq w(e')$ with $w(e') = 2\alpha g$ by the same argument as in the former case, meaning $2g \leq \frac{w_{\text{opt}}}{\alpha}$. That is $w(M) \leq w(p, c) + \sum_{e \in E, e \notin \{(p,c), e'\}} w(e)$ as $e' \notin M$ because M is a matching and cannot have two edges sharing a common node. Thus $w(M) \leq \varepsilon + g < 2g$, hence $w(M) < \frac{w_{\text{opt}}}{\alpha}$ and thus $\mathcal{A}$ is not α-approximate in that case as well. □

Proposition 1 essentially tells us that without additional assumptions, we need to compute at least m' weights and in this case, we can run the optimal algorithm on G' and add all isolated edges afterwards which obviously produces the optimal solution for G (in the rest of the paper we will implicitly assume that G does not have any isolated edges). To circumvent the impossibility and aim to compute less than m' weights, we assume that there exists an oracle that provides us with the vertices of P and possibly of C in an order σ_P (or σ_C) which guarantees additional properties about the weights. The matching algorithm $\mathcal{A}$'s aim is to heuristically use σ_P and σ_C to query the weight $w(p, c)$ of particular edges $(p, c) \in E$. More generally, one may assume that the oracle is powerful enough to provide the edges that the matching algorithm should consider in an order σ_E over E so that edges with higher weights are generally considered earlier on. In such a case, observe that for any given matching algorithm $\mathcal{A}$, there exists an optimal order $\sigma_{\mathcal{A}}$ over E that optimizes the weight of the matching produced by $\mathcal{A}$ (note, for some algorithm, all such orders may still produce the same result). Since our goal is to design efficient matching algorithms that minimize the number of weight calculations, we cannot assume that edges are processed by $\mathcal{A}$ in the order $\sigma_{\mathcal{A}}$ but rather the goal is to design an algorithm $\mathcal{A}'(\sigma_P, \sigma_C)$ which produces a matching of bounded approximation ratio given the oracle's orders, while calculating a hopefully limited number of weights. Assuming there exist heuristic orders on P and C with interesting properties on the weight function stems from the settings of our original motivating problems of peer matching among energy communities [7]. In the next section, we design greedy algorithms exploiting σ_P and σ_C and show their approximation ratio. Our aim is to assume that σ_P and σ_C entail weak properties on the weights but strong enough to be able to reach a sub-quadratic number of weight calculations in $\max\{|P|, |C|\}$ while keeping a bounded approximation ratio for the calculated matching.

3. Discovery Algorithms for the One-to-One Assignment Problem

3.1. The Greedy-Local algorithm

We first study the following greedy algorithm: the vertices of the set P are processed one-by-one in the oracle's order σ_P where σ_P was designed to have earlier vertices more likely to have high gains for a given task than vertices appearing later in the order. Each time a node is processed, its full neighborhood is examined and the available edge with highest weight is selected to be added to the matching. In the following, during the round where $p \in P$ is considered, we refer to an edge (p, c) as being *available* if the endpoint $c \in C$ of the edge has not been previously *blocked* by adding another edge (p', c) to the matching at an earlier stage of the algorithm (which is greedy and never reconsiders previous choices). We show that this *Greedy-Local algorithm* achieves a bounded approximation ratio if $\sigma_P = p_1, \ldots, p_n$ orders the vertices in P such that for any $1 \leq i < j \leq n$, the weight of (p_j, c) is bounded by β times the weight of (p_i, c). Since in our original application, many of the weights of the input graph may be equal to each other, the "best value" that β may take is 1 (i.e. any subsequent edge sharing an endpoint in C with an edge being processed has either an equal or strictly smaller weight); hence, for simplicity, we assume hereafter $\beta \geq 1$. Note that without any additional assumptions, Alg. 1 does not produce a bounded approximation in general as its greedy decisions do not take the "future" into consideration, hence adding (p_i, c) to the matching might remove the possibility to add a later-to-be-processed (p_j, c), with $j > i$, and whose weight might be arbitrarily large.

Assumption 1. *Assume that P is processed in the order $\sigma_P = p_1, p_2, \ldots, p_n$, so that for any $p_i, p_j \in P$ with $i, j \geq 1$, $i < j$ and $c \in C$ such that $(p_i, c) \in E$ and $(p_j, c) \in E$, we have $w(p_j, c) \leq \beta \cdot w(p_i, c)$, with $\beta \geq 1$.*

Proposition 2. *Under Assumption 1, Alg. 1 has approximation ratio at most $1 + \beta$.*

Proof. Let M be the matching obtained by an optimal algorithm and M' the one by Alg. 1. Let $f : M \to M'$ be a function that projects the edges selected by the matching M onto the edges of M' defined as follows. (1) For $e \in M$, if $e \in M'$, then $f(e) = e$. (2) For $e = (p_j, c) \in M$ and $e \notin M'$, consider the two following cases. (a) At the beginning of p_j's turn, e was not selected in M' because it was already *blocked*. That is, e was not among the available edges considered by Alg. 1 during p_j's turn, and since p_j has not been assigned to any node in C yet, that means there exists a blocking edge $(p_i, c) \in M'$ with $i < j$ that has been added to M' before p_j's round. Define $f(e) = (p_i, c)$ then. (b) The complementary case is that e was not selected in M' during p_j's turn but it was still available to pick (that is, e was not blocked). In this situation, Alg. 1 picks the edge with highest weight locally and since $(p_j, c) \notin M'$ there must

Alg. 1: Greedy-Local Matching

```
   Input  : A bipartite graph G = (P ∪ C, E) with sets
            P = p_1, p_2, ..., p_n and C = c_1, c_2, ..., c_q
   Output: M, a matching of E;
   // Initialization
1  M ← ∅ ;
2  foreach j ∈ C do
3  |  available_j ← True ;
   // Greedy Matching Loop
4  for 1 ≤ i ≤ n do
5  |  N ← {1 ≤ j ≤ q | {p_i, c_j} ∈ E ∧ available_j} ;
6  |  if N ≠ ∅ then
7  |  |  if |N| > 1 then
8  |  |  |  foreach j ∈ N do
9  |  |  |  |  b_j ← weight(p_i, c_j);
10 |  |  |  j ← arg max_{j∈N} b_j;   // In the case
   |  |  |     of a tie, take the smallest j
11 |  |  else
12 |  |  |  j ← N[1]; // Retrieve the first
   |  |  |     value
13 |  |  available_j ← False ;
14 |  |  M ← M ∪ {p_i, c_j};
15 return M;
```

Alg. 2: Naive-Local Matching

```
1  for 1 ≤ i ≤ n do
2  |  N ← {1 ≤ j ≤ q | {p_i, c_j} ∈ E ∧ available_j};
3  |  if N ≠ ∅ then
4  |  |  j ← N[1] ;
5  |  |  available_j ← False; M ← M ∪ {p_i, c_j};
6  return M;
```

Alg. 3: ℓ-Greedy-Local Matching

```
1  for 1 ≤ i ≤ n do
2  |  N ← {1 ≤ j ≤ q | {p_i, c_j} ∈ E ∧ available_j} ;
3  |  if N ≠ ∅ then
4  |  |  if |N| > 1 then
   |  |  |  // Keep the ℓ + 1 first values
5  |  |  |  N ← N[: ℓ + 1] ;
6  |  |  |  foreach j ∈ N do
7  |  |  |  |  b_j ← weight(p_i, c_j);
8  |  |  |  j ← arg max_{j∈N} b_j; // As in Alg. 1, line 10
9  |  |  else
10 |  |  |  j ← N[1] ;
11 |  |  available_j ← False ; M ← M ∪ {p_i, c_j};
12 return M;
```

All 3 algorithms have the same input/output (as Alg. 1) and Alg. 2 and Alg. 3 start by the same initialization lines (1-3) as in Alg. 1.

be another edge $(p_j, c') \in M'$ with $c' \neq c$ with a higher weight that has been selected instead. Define $f(e) = (p_j, c')$ in this case.

By exhaustion of possible cases, every edge of M has an image in M'. We now prove that every edge $(p_i, c) \in M'$ has at most two preimages under the function f. If $(p_i, c) \in M$, the edge has only itself as preimage as this implies that there exist no edges in M' such that $(p', c) \in M'$ with $p' \neq p_i$ nor $(p_i, c') \in M'$ with $c' \neq c$ as M' is a matching of the edges; in this case, f is prevented from applying Cases 2a and 2b and only the Case 1 remains. Now, consider $(p_i, c) \notin M$. We show that there is only a single edge $e \in M$ such that Case 2a applies so that $f(e) = (p_i, c)$, and the same for Case 2b. For Case 2a to apply, e must be of the form (p_j, c) with $j > i$ and since M is a matching there cannot exist another edge in M containing node c. Similarly, for Case 2b to apply, we need to have $(p_i, c') \in M$ and for the same reason there cannot be another edge in M sharing the node p_i.

Note that in Case 1, we have trivially $w(e) \leq w(f(e))$; in Case 2a, we have $w(e) \leq \beta \cdot w(f(e))$ by direct application of Assumption 1; in Case 2b, we have $w(e) \leq w(f(e))$ as the algorithm chooses $f(e)$ as the local maximum of unblocked edges and both e and $f(e)$ are then unblocked. Hence, we can now bound the total weight of the matching M as:

$$w(M) = \sum_{e \in M} w(e) \leq \sum_{e \in M,\, f(e)=e} w(f(e)) \;+\; \sum_{e=(p_j,c) \in M,\, f(e)=(p_i,c),\, i<j} \beta \cdot w(f(e)) \;+\; \sum_{e=(p_j,c) \in M,\, f(e)=(p_j,c'),\, c \neq c'} w(f(e)) \,.$$

As each edge $e' \in M'$ appears either only in the first sum, or at most once in each of the last previous two sums (see discussion above): $w(M) \leq \sum_{e' \in M' \mid \exists e \in M, f(e)=e'} (1+\beta) \cdot w(e')$. As all weights are greater than zero, we get $w(M) \leq \sum_{e' \in M'} (1+\beta) \cdot w(e') \leq (1+\beta) \cdot w(M')$. □

We note that this first algorithm may reduce drastically the number of computed weights, as blocked choices as well as vertices left with a single choice are not computed during the execution. However, in the worst case, the algorithm does end up computing all weights in G. The example input given in Fig. 2 illustrates that there exist instances where Alg. 1 has an approximation ratio of at least $1 + \beta - \varepsilon$ under Assumption 1, where $\varepsilon > 0$ is only introduced to break

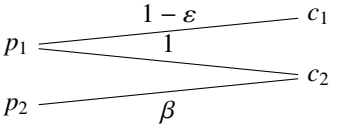

Fig. 2. Counter-example for Alg. 1, $\varepsilon > 0$.

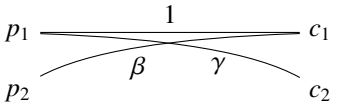

Fig. 3. Counter-example for Alg. 2.

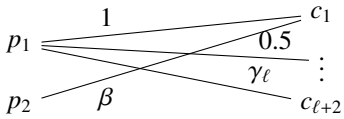

Fig. 4. Counter-example for Alg. 3.

ties. Note that one can introduce the following symmetric assumption in comparison to Assumption 1 reversing the sets P and C.

Assumption 2. *Assume the set C is provided in the order $\sigma_C = c_1, c_2, \ldots, c_q$, so that for any $c_i, c_j \in C$ with $i, j \geq 1$, $i < j$ and $p \in P$ such that $\{p, c_i\} \in E$ and $\{p, c_j\} \in E$, we have $w(p, c_j) \leq \gamma \cdot w(p, c_i)$, with $\gamma \geq 1$.*

If we run Alg. 1 with input $G = (C \cup P, E)$, i.e., inverting the set P and the set C in its input, Assumption 2 entails that the output is a $(1 + \gamma)$-approximation over $G = (P \cup C, E)$ by following Proposition 2 with $\beta = \gamma$. Observe that Alg. 1 is not symmetric in P and C and the output that is produced is naturally different than with the original input.

Proposition 3. *Under Assumption 1, there exist instances where Alg. 1 has an approximation ratio of at least $1 + \beta - \varepsilon$ for any $\varepsilon > 0$.*

Proof. Let us consider the example input given by Fig. 2 with $P = \{p_1, p_2\}$ and $C = \{c_1, c_2\}$. Assumption 1 holds on this input as we have $w(p_2, c_2) \leq \beta \cdot w(p_1, c_2)$ and this is the only pair of edges where it can apply. Alg. 1 selects as matching the pair (p_1, c_2) as it is the local maximum of p_1, with weight 1, to compare with the optimal matching which selects the two other edges with total weight $1 + \beta - \varepsilon$. □

3.2. The Naive-Local algorithm

The introduced assumptions allow to make greedy choices during the processing of nodes by the matching algorithm, however, they do not always guarantee that one can omit the computation of the weight of at least one edge of the input graph if the assumptions are used separately. For instance for Assumption 1, consider a graph where each p_i is connected to c_{2i} and c_{2i+1} and nothing else, hence omitting the computation of a single weight of the graph may lead to an unbounded approximation as Assumption 1 does not provide bounds on the omitted weight. Observe that the same argument applies in a symmetric manner with Assumption 2. However, if both previously introduced assumptions hold in the oracle's orders σ_P and σ_C, then one can actually design an algorithm computing no weights at all but achieving a bounded approximation of the optimal matching. Following a similar proof as in Proposition 2, one can derive that if both assumptions hold, then Alg. 2 produces a $(\beta + \gamma)$-approximate matching without calculating any weights of the input. Using the example of Fig. 3, one can also show that any matching algorithm that calculates 0 weights (and in particular Alg. 2) cannot be better than $(\beta + \gamma)$-approximate under both assumptions combined.

Proposition 4. *If Assumptions 1 and 2 hold, Alg. 2 produces a $(\beta + \gamma)$-approximate matching without calculating any weights.*

Proof. The proof follows the same structure to the one of Proposition 2. The difference is only that the Naive-Local algorithm assigns the first unblocked edge (in C's provided order σ_C) to p_j whereas the Greedy-Local algorithm chooses the local maximum of the unblocked edges. Hence, we can define similarly f and we have again that any edge of M' can only be the image by f of at most two different preimages. By using the same arguments, the same inequalities on weights hold for Cases 1 and 2a. Observe now that in Case 2b with $e = (p_j, c_x) \in M$ and $f(e) = (p_j, c_y) \in M'$, we have $x > y$ as c_y is chosen by M' as the first available edge, hence, we have that $w(e) \leq \gamma \cdot w(f(e))$ following Assumption 2. With analogous concluding arguments, we get that each edge of M' can either be also present in M and has then a unique image by f, or appear at most once in each Case 2a and 2b, entailing: $w(M) \leq \sum_{e' \in M' \mid \exists e \in M, f(e) = e'} (\gamma + \beta) \cdot w(e') \leq (\gamma + \beta) \cdot w(M')$. □

Proposition 5. *Under Assumptions 1 and 2, any matching algorithm that calculates 0 weights cannot be better than $(\beta + \gamma)$-approximate.*

Proof. Let us consider the 4-nodes instance given by Fig. 3. Given the provided ordering of vertices in P and C, we have that both Assumptions 1 and 2 hold on the instance. Note first that the Naive-Local matching on this instance produces $\{(p_1, c_1)\}$ with weight 1 whereas the optimal picks the two other edges with weight $\beta + \gamma$. Now, consider a matching algorithm $\mathcal{A}$ that picks (p_1, c_2) and (p_2, c_1). In that case, change the instance so that $w(p_1, c_1) = \alpha$ with α arbitrarily large and all other weights set to $\varepsilon > 0$ to simplify (note that both our underlying assumptions hold in this situation as well). $\mathcal{A}$ is then arbitrarily far from the optimal matching that selects (p_1, c_1). □

3.3. The ℓ-Greedy-Local algorithm

Our first results show that the first set of assumptions that was considered may be unsatisfactory for two reasons: either one of the assumptions holds and all weights may end up being computed or both assumptions hold at the same time and absolutely no weight calculations are required to reach a bounded approximation ratio. This may indicate that the assumptions could be too strong in some sense. We design here weaker assumptions that only require the condition on one set to hold (e.g., Assumption 1) and a weaker form of the other assumption: the bound holds between node $p \in P$ and $c, c' \in C$ if there exist at least ℓ other neighbors of p between c and c' when taken in σ_C's order. That is, we do not control the weight of successive edges in a given node's neighborhood but if there are ℓ other edges (p, c^j) between two edges (p, c^0) and $(p, c^{\ell+1})$, then the latter one must have a bounded weight in comparison to (p, c^0). The following assumption allows us to design a matching algorithm (Alg. 3) requiring only at most $\ell + 1$ weight computations for each node in P. In the following assumption, smaller values for ℓ make the assumption stronger, with $\ell = 0$ being equivalent to Assumption 2 (hence $\gamma_0 = \gamma$) and $\ell = q$ being always true for any input.

Assumption 3. *Assume $\ell \geq 0$ and $\sigma_C = c_1, c_2, \ldots, c_q$, so that for any $c_i, c_j \in C$ with $i, j \geq 1$, $i < j$ and $p \in P$ such that $(p, c_i), (p, c_j) \in E$ and such that $|\{(p, c_x) \in E \mid i < x < j\}| \geq \ell$, we have $w(p, c_j) \leq \gamma_\ell \cdot w(p, c_i)$, , with $\gamma_\ell \geq 1$.*

Proposition 6. *Under Assumptions 1 and 3, Alg. 3 has approximation ratio at most $\beta + \gamma_\ell$.*

Proof. The proof follows the same structure as the one for Proposition 2. Define M as an optimal matching, M' as the matching produced by Alg. 3 on G, and f as a mapping of M's edges into M' similarly and have identical Cases 1 and 2a. For Case 2b, that is when we consider an edge $e = (p, c) \in M$ such that $e \notin M'$ while considering that (p, c) is unblocked during p's assignment round, we define f as the edge with the maximum weight among the $\ell + 1$ first unblocked edges (in the same way as Alg. 3 picks the edge during p's round). Since for each p, we assign as before an edge of its neighborhood by f, our previous arguments hold regarding the number of preimages by f. Now, consider the bound on the weight of edges in M. We know that $w(e) \leq w(f(e))$ in Case 1 (trivial) and $w(e) \leq \beta \cdot w(f(e))$ in Case 2a thanks to Assumption 1.

In Case 2b, let's consider two possible subcases. (1) If there are at most $\ell + 1$ unblocked edges during p's round, then since e is unblocked, it is among these edges. Hence, by the property that $w(f(e))$ is the maximum of the weights of the unblocked edges, we get $w(e) \leq w(f(e))$.

(2) Suppose there are strictly more than $\ell + 1$ unblocked edges. Since if e were among the first $\ell + 1$ ones we would also have $w(e) \leq w(f(e))$, let's assume $e = (p, c_j)$ is not among these edges. By Assumption 3, recall that one cannot bound the weights of the edges between p and its neighbors c_i such that (p, c_i) is among the ℓ distinct edges incident to p directly preceding (p, c_j) in σ_C's order; note $T(c_j)$ the set of these edges. If Alg. 3 selects an edge $f(e) = (p, c_x)$ outside $T(c_j)$, we can apply the aforementioned assumption and get $w(e) \leq \gamma_\ell \cdot w(f(e))$. Let's suppose thereafter that Alg. 3 selects an edge $(p, c_x) \in T(c_j)$. Observe that among the ℓ edges of $T(c_j)$, some of them might be blocked and others unblocked, however, in any case there exists at least one unblocked edge $(p, c') \notin T(c_j)$ because $|T(c_j)| = \ell$ and we assumed at least $\ell + 2$ unblocked edges. Note that there are at most $\ell - 1$ edges between (p, c') and (p, c_x) in σ_C's order, hence $w(p, c')$ was considered by the algorithm during p's round. Finally, we have $w(p, c') \leq w(p, c_x)$ because the algorithm picked the edge with the best weight, and thus $w(p, c_j) \leq \gamma_\ell \cdot w(p, c')$ by application of Assumption 3, which gives us $w(e) \leq \gamma_\ell \cdot w(f(e))$ here as well.

By reusing similar arguments as in the previous proofs, we get $w(M) \leq (\beta + \gamma_\ell) \cdot w(M')$. □

Proposition 7. *Under Assumptions 1 and 2, Alg. 3 is $(\beta + \gamma)$−approximate.*

Proof. Follows directly from Proposition 6 when $\ell = 0$. □

The example of Fig. 4 can be used to show that under Assumptions 1 and 3, Alg. 3 has approximation ratio at least $\beta + \gamma_\ell$. By symmetry, we use Assumption 9 to obtain symmetric results.

Proposition 8. *Under Assumptions 1 and 3, Alg. 3 has approximation ratio at least $\beta + \gamma_\ell$.*

Proof. We use the example of Fig. 4 where p_1 has $\ell+2$ neighbors, and on which Assumption 1 only applies to (p_1, c_1) versus (p_2, c_1) and Assumption 3 to (p_1, c_1) versus $(p_1, c_{\ell+2})$. In the example, the algorithm picks (p_1, c_1) for a weight of 1 whereas the optimal matching picks $(p_1, c_{\ell+2})$ and (p_2, c_1) for a weight of $\beta + \gamma_\ell$. □

Assumption 4. *Assume $\sigma_P = p_1, p_2, \ldots, p_n$, so that for any $p_i, p_j \in P$ with $i, j \geq 1$, $i < j$ and $c \in C$ such that $(p_i, c), (p_j, c) \in E$ and such that $|\{(p_x, c) \in E \mid i < x < j\}| \geq \ell$, we have $w(p_j, c) \leq \beta_\ell \cdot w(p_i, c)$, with $\beta_\ell \geq 1$.*

Proposition 9. *Under Assumptions 2 and 4, Alg. 3 has approximation ratio at most $\gamma + \beta_\ell$.*

At last, we explain briefly why lifting Assumption 1 controlling the order in which vertices of P are processed and replacing it by a bounded variant tolerating out-of-orderness such as Assumption 4 leads to impossibility to approximate the optimal matching. One can consider a path as an instance and can derive that any algorithm only inspecting a bounded number of edges $f(\ell)$ to decide greedily which edge to add will have no control on the weight of the edges connected to the last inspected edge on the input path. We also note that Assumption 4, for $\ell \geq 1$, does not imply any bounds between the weights of (p, c_1) and (p, c_2), hence any algorithm that does not check all weights has unbounded approximation ratio.

Proposition 10. *Under Assumptions 3 and 4 with $\ell \geq 1$, suppose a greedy matching algorithm $\mathcal{A}$ decides to add a particular edge (p, c) after examining a bounded number of edges $f(\ell) < |E|$. Then $\mathcal{A}$ does not provide a bounded approximation ratio.*

Proof. Without loss of generality, we can consider as a counter example a path of length $f(\ell) + 1$ of the form $p_1, c_1, p_2, \ldots, x_{f(\ell)}, y_{f(\ell)+1}$. After executing $\mathcal{A}$, the weight of the edge $(x_{f(\ell)}, y_{f(\ell)+1})$ was not checked though it can be arbitrarily large because if $x_{f(\ell)}$ has only two neighbors, then both assumptions (about β_ℓ and γ_ℓ) do not apply to that particular edge and its weight can thus be arbitrarily large in this case. □

4. Extensions and Conclusions

One can reduce any one-to-many assignment problem (that is, where $p \in P$ may be allowed to be paired with up to k consumers of C) to a one-to-one assignment problem in the following manner. For each $p \in P$, make k copies $p^1, \ldots, p^k$ of node p, while keeping the original weights, i.e., $\forall c \in C, w(p^j, c) = w(p, c)$. Finally, solve the maximum matching in bipartite graphs problem with the input $P' = \{p^j \mid j \in [1..k], p \in P\}$ and C. This reduction allows to extend all our results to one-to-many assignment problems with all bounds shown in Table 1 still holding the same way using the algorithms we have introduced in this work. They also extend to the bipartite hypergraph matching whenever the weight of a hyperedge can be upper- and lower-bounded by the sum of the pairwise weights. We note that for a given input, processing order and with access to all weights, one can easily compute the values of β, γ, β_ℓ and γ_ℓ which can become good estimations for bounds on larger instances in a given application. We have in this work extended the notion of discovery algorithms to assignment problems, and we believe our work provides useful theoretical bounds for algorithms that are efficient in practice[1]. Our findings open up for further rehabilitation of greedy algorithms in theoretical analysis, and advocate that greedy algorithms do not only often provide computationally-efficient solutions to hard problems but can also be formally analysed within the scope of concrete applications.

[1] Using data from [7] (search radius of 1km that is $n = 445$, $q = 1776$, $m = 9903$) and the order of resources for the prosumers (members of set P) and consumers (set C), we get $\beta = 3.10$, $\beta_2 = 1.95$, $\beta_{17} = 1$, $\gamma = 4.15$, $\gamma_{20} = 1.95$ and $\gamma_{58} = 1$. Greedy-Local computes 85.8% of the weights and achieves 98.2% of the optimal weight W_{opt}, Naive-Local 94.4% of W_{opt}, and e.g. ℓ-Greedy-Local computes 8.5% of the weights for 96.4% of W_{opt} with $\ell = 1$, and 39% of the weights for 98% of W_{opt} for $\ell = 10$.

5. Acknowledgements

We would like to thank the anonymous reviewers for their precise, relevant and valuable feedback on an earlier draft of this work.

Ralf Klasing is partially supported by the ANR project TEMPOGRAL (ANR-22-CE48-0001). Romaric Duvignau is partially supported by the TANDEM project within the framework of the Swedish Electricity Storage and Balancing Centre (SESBC), funded by the Swedish Energy Agency together with five academic and twenty-six non-academic partners.

References

[1] H. W. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistics Quarterly 2 (1-2) (1955) 83–97.
[2] L. Ramshaw, R. E. Tarjan, On minimum-cost assignments in unbalanced bipartite graphs, HP Labs, Palo Alto, CA, USA, Tech. Rep. HPL-2012-40R1.
[3] R. Duan, S. Pettie, Linear-time approximation for maximum weight matching, Journal of the ACM (JACM) 61 (1) (2014) 1–23.
[4] C. Szepesvári, Shortest path discovery problems: A framework, algorithms and experimental results, in: D. L. McGuinness, G. Ferguson (Eds.), Proc. 19th National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, 2004, San Jose, California, USA, AAAI Press / The MIT Press, 2004, pp. 550–555.
[5] C. T. Caro, J. Doncel, O. Brun, Optimal path discovery problem with homogeneous knowledge, Theory Comput. Syst. 64 (2) (2020) 227–250.
[6] R. Duvignau, V. Heinisch, L. Göransson, V. Gulisano, M. Papatriantafilou, Benefits of small-size communities for continuous cost-optimization in peer-to-peer energy sharing, Applied Energy 301 (2021) 117402.
[7] R. Duvignau, V. Gulisano, M. Papatriantafilou, Efficient and scalable geographical peer matching for p2p energy sharing communities, in: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, 2022, pp. 187–190.
[8] B. Chandra, M. M. Halldórsson, Greedy local improvement and weighted set packing approximation, Journal of Algorithms 39 (2) (2001) 223–240.
[9] P. Berman, A $d/2$ approximation for maximum weight independent set in d-claw free graphs, in: Scandinavian Workshop on Algorithm Theory, Springer, 2000, pp. 214–219.
[10] M. Neuwohner, Passing the limits of pure local search for weighted k-set packing, in: N. Bansal, V. Nagarajan (Eds.), Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, 2023, SIAM, 2023, pp. 1090–1137.
[11] T. Erlebach, M. Hoffmann, D. Krizanc, M. Mihalák, R. Raman, Computing minimum spanning trees with uncertainty, in: S. Albers, P. Weil (Eds.), STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, 2008, Proceedings, Vol. 1 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2008, pp. 277–288.
[12] T. Erlebach, M. Hoffmann, F. Kammer, Query-competitive algorithms for cheapest set problems under uncertainty, Theor. Comput. Sci. 613 (2016) 51–64.
[13] I. D. Aron, P. Van Hentenryck, On the complexity of the robust spanning tree problem with interval data, Operations Research Letters 32 (1) (2004) 36–40.
[14] A. Kasperski, P. Zieliński, An approximation algorithm for interval data minmax regret combinatorial optimization problems, Information Processing Letters 97 (5) (2006) 177–180.
[15] H. Yaman, O. E. Karaşan, M. Ç. Pınar, The robust spanning tree problem with interval data, Operations research letters 29 (1) (2001) 31–40.