Geographical Peer Matching for P2P Energy Sharing

Romaric Duvignau¹, Vincenzo Gulisano¹, and Marina Papatriantafilou¹

¹Chalmers University of Technology (CTH), Sweden {duvignau,vinmas,ptrianta}@chalmers.se

December 22, 2021

Abstract

Significant cost reductions attract ever more households to invest in small-scale renewable electricity generation and storage. Such distributed resources are not used in the most effective way when only used individually, as sharing them provides even greater costsavings. Energy Peer-to-Peer (P2P) systems have been shown to be beneficial for prosumers and consumers through reductions in energy cost while being attractive to grid or service provider. However, many practical challenges have to be overcome before all players could gain in having efficient and automated local energy communities, including the inherent complexity of matching together geographically distributed peers and the significant computation required to calculate the local matching preferences. Hence dedicated algorithms are required to be able to perform a cost-efficient matching of thousands of peers in a computationalefficient fashion. We define and analyse in this work a precise mathematical modelling of the geographical peer matching problem. We then propose and analyse dedicated algorithms. Our experimental study, based on real-world energy data, demonstrates that our solutions are both efficient in terms of cost-savings achieved by the peers and in terms of computing requirements. Previous results have already shown that small communities were efficient but only small-enough datasets were used allowing for brute-force solutions. Our scalable algorithms thus provide one core building block on the way to build fully practical and data-efficient peer-to-peer energy sharing communities for large-scale systems.

1 Introduction

Renewable electricity generation is becoming more affordable to end-users as the initial investment cost has been drastically cut, thus transforming the traditional residential households from consumers into prosumers [32] (capable of producing locally their own electricity). Sharing resources (for example solar PV panels and battery systems) in Peer-to-Peer (P2P) setups can be leveraged as a way to optimize the cost-benefits from distributed resources. Hence, primarily driven by

the interest to reduce even more their cost, there is interest in P2P energy communities [16], a high-potential concept that caught up the attention of the research community in the recent years [34]. The principle of P2P energy sharing is for different end-users to share their resources locally in small or larger groups, in order to reduce their energy bill. Locally in each community, energy is either "traded" at regular intervals (e.g. every hour) or "exchanged for free" with a later gratification scheme. P2P energy sharing bypasses the centralized grid by letting the peers cooperate in a distributed fashion in order to share in the best way their energy resources. Hence, to lower their cost, households are encouraged to use as much as possible the electricity that is generated locally instead of the one from the grid, with local benefits due to reduced tax fees, as well as an increase in local self-consumption.

However, organizing end-users into communities reveals to be a challenging task. Short-term communities (lasting for e.g. 10min) are formed by using usually game-theoretic approaches [33] and only taking into account the current state of the system (e.g. amount of electricity being produced and consumed by the different end-users, as well as market price). Forming longterm communities (used over months or years) is an ever more rewarding and challenging task. Chau et al [5] have investigated stable partitioning, i.e., a given partitioning can be rejected if any group of peers would gain more by forming a different one, similar to the stable marriage [25] problem for pairs. Zhou et al [39, 4] expand the partitioning to communities of size k > 2and evaluate partition-forming algorithms for groups of size 2 or 3 over a set of 30 households. Duvignau et al [10, 11] advocate small-scale communities made of a few peers only (2 to 5) claiming that smaller groups are both efficient in terms of data and cost. No mathematical analysis is done concerning the cost of computing the different matchings as the work is essentially focused on energy cost-optimization and data-efficiency. In all the aforementioned works, the computed long-term partitions in the experimental study do not involve a high number of nodes (100 at maximum) or larger neighborhoods than those of size 3, nor do they use information about the geographical positions of the nodes.

Motivations Current state-of-the-art [18, 4, 11] relies on exhaustive search to compute the optimal solution from datasets containing only a small number of nodes. To scale up towards large systems, we introduce the *Geographical Peer Matching* (GPM) problem that consists in forming the energy communities based on both geographical information about the peers as well as their local matching preferences. This provides a natural way of reducing the search space but as we show in this work, dedicated algorithms are required to cope with the computational complexity of the GPM problem.

Challenges and Research Questions The main challenges in establishing efficiently P2P energy sharing communities are threefold: (i) peers continuously produce data and have limited knowledge of how their future local data will look like, (ii) computing peers' optimization options for preferences requires both access to the relevant data and the execution of a computationally expensive procedure and (iii) peers should favor getting matched with geographically neighboring peers to reduce transmission losses and the impact on the underlying infrastructure. In particular, weights in the matching (i.e., the local matching preferences) are not provided beforehand but must be computed on the fly and this requires additional communication between the participants. An additional difficulty stems from allowing the formation of groups of size 3 and above as the matching problem becomes NP-hard in this case (as shown in Section 3) and thus in practical systems, it is not feasible to do exhaustive searches any longer.

This raises the following research questions: (1) Can one translate the challenges to a formal model that can capture the benefits and complexity for the prosumers? (2) How do the maximum size and geographical diameter for communities influence the cost-efficiency of the peer matching? (3) Is it possible to design matching algorithms that are both cost-efficient and scalable?

Contributions We present concrete matching mechanisms capable of assigning consumers to prosumers to form P2P energy sharing communities. One core contribution is the presentation of a mathematical modelling of the GPM problem expressed as finding a maximum weight bipartite partitioning in hypergraphs. We further propose and analyse different algorithms that solve the GPM problem. We study the cost-efficiency of the different algorithms based on an experimental study involving consumption data from 2221 real households and realistic solar profiles (both over a year), and using a realistic distribution of renewable energy resources among the peers. Our findings highlight that our algorithms for P2P energy sharing communities are both cost-efficient (providing significant cost-savings to all peers) and scalable (being able to match at least thousands of users).

The paper is organized as follows. Section 2 presents in more detail P2P energy sharing and cost-

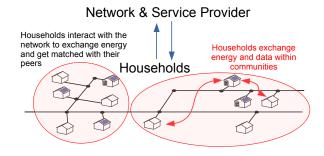


Figure 1: Overview of P2P energy sharing showing grouping and interactions between prosumers (equipped with PV panels on roof top and optionally a battery system) and traditional consumers (without any energy resources).

optimization of distributed resources in this context. Section 3 presents our mathematical modelling of the GPM problem and some of its variants. In Section 4 we present our algorithms and analyse their computational overhead. In Section 5, we present an extensive experimental study of the performance of the algorithms based on electricity data from real-world households. The section following that discusses related work, whereas Section 7 presents our conclusions.

2 System Model

We define here the P2P energy sharing communities notion, its underlying assumptions and consequences. We then define how to optimize peers cost in such a context and present how this translates into a matching problem when several communities are considered.

2.1 P2P Energy Sharing

Let us recall the basic requirements of the traditional energy infrastructure, here not considering other forms of energy other than electricity for the purpose of simplicity. Consumers must match their consumption by importing from the grid their electricity demand, whereas prosumers use in priority their local generation (e.g. PV panels); in case of surplus, it is sold to the grid, while in the opposite situation, prosumers must import electricity from the grid. The situation complicates as soon as a prosumer is equipped with both electricity production and storage, i.e., needing online decisions whether to store surplus or sell it to the grid, and whether to use stored electricity or rather buy it from the grid.

Set in the context of increasing decentralization of the energy infrastructure, P2P energy sharing (see Figure 1) consists in forming local energy communities of cooperative end-users. The goal is to make the most of the distributed resources and hence achieve even greater cost reductions. However, this means that participants need now to consider also the state and decisions of the other actors in order to coordinate and optimize the benefits of their local resources. Coordination takes place at regular time intervals (e.g. 1 hour), where any consumption within a community can be offset by importing the equivalent amount of electricity from a peer. Such an exchange can get instantly gratified leading to a local trading market [30, 38]. Since there is an inherent infrastructure cost to allow and maintain energy exchanges between a large number of end-users, exchanges have been restrained to occur within independent long-term communities of end-users [22, 18, 4, 11] that can last for month(s) or year(s). At the end of predetermined billing periods, each user pays an electricity bill taking into account all exchanges that occurred within the said period.

2.2 Single-user Cost-Optimization

The cost-optimization problem consists in minimizing the yearly electricity bill for a particular end-user based on locally available electricity data: amount of consumption, generation and price. In order to minimize the cost, we adopt the following LP-formulation cost-optimization following similar models used in the recent literature (cf. [5, 18, 11]). The electricity cost $\mathbf{cost}(h, t)$ of the end-user h at hour t, expressed in $\mathbf{\in}$, is assumed to be as follows:

$$\mathbf{cost}(h,t) = e\mathbf{l_{in}}(h,t) \cdot (\mathbf{price}(t) \cdot (1+\mathbf{tax}) + \mathbf{el_{tax}}) - e\mathbf{l_{out}}(h,t) \cdot (\mathbf{price}(t) + \mathbf{el_{net}}),$$

where

- $el_{in}(h,t)$ is the amount of electricity in kWh bought from the grid by h at hour t;
- price(t) is the price of electricity in €/kWh at hour t, assumed independent of h;
- tax is a relative tax level, \mathbf{el}_{tax} is a fixed electricity tax in \in /kWh added on top of market price, and \mathbf{el}_{net} is a small payment in \in /kWh for selling electricity to the grid;
- $el_{out}(h,t)$ is the amount of electricity in kWh sold to the grid by h at hour t.

We assume $\mathbf{el}_{net} < \mathbf{el}_{tax}$ and $\mathbf{tax} \geq 0$, and we note $\mathbf{el}_{\text{CONS}}(h,t)$ the consumption (or electricity demand) of user h for hour t. Consumers do not have any resources, hence the yearly cost is obtained directly from their consumption, that is $\mathbf{el}_{in}(h,t) = \mathbf{el}_{\text{CONS}}(h,t)$ for all hours t. Prosumers with only electricity generation but no storage have always interest to use in priority their local production to avoid to pay tax on electricity coming from the grid; hence, they optimize their cost by setting

$$\boldsymbol{el_{in}}(h,t) = \begin{cases} x_{h,t}, & \text{if } x_{h,t} > 0, \\ 0, & \text{otherwise;} \end{cases} \quad \boldsymbol{el_{out}}(h,t) = \begin{cases} -x_{h,t}, & \text{if } x_{h,t} < 0, \\ 0, & \text{otherwise;} \end{cases}$$

with the electricity balance $x_{h,t} = \mathbf{elgen}(h,t) - \mathbf{elcons}(h,t)$ where $\mathbf{elgen}(h,t) = \mathbf{PV}_h \cdot \boldsymbol{sun}(t)$ is the electricity generated by h during hour t (in kWh) based on h's PV capacity \mathbf{PV}_h and the sun's intensity at hour t. Prosumers having both electricity generation and storage can optimize their cost over a period of time

from t_0 to t_r through running a LP solver of the following formulation:

- Objective function: minimize bill $(h, [t_0, t_r]) = \sum_{t=t_0}^{t_r} \mathbf{cost}(h, t)$.
- Constraints (for all $t_0 \le t \le t_r$): $0 \le bat(h,t) \le \mathbf{B}_h$, and $bat(h,t) = bat(h,t-1) + x_{h,t} + el_{in}(h,t) el_{out}(h,t)$.
- Optimization variables: $\{bat(h,t), el_{in}(h,t), el_{out}(h,t) \mid t_0 \leq t \leq t_r\}.$

with $bat(h, t_0 - 1)$ indicating the initial battery level and \mathbf{B}_h the battery capacity of h (in kWh).

2.3 Community Cost-Optimization

One may wonder how the end-users may in our context make the most of their distributed resources. Instead of fixing one of the many forms of a local trade market, we consider that the energy exchanges occur "for free" within the community to obtain and analyse the lowest achievable cost as a community, while postponing the billing of individual exchanges to the end of the billing period. This way, communities may reach the "best obtainable" gain while settling individual gain by a subsidiary mechanism at the end of the billing period (not analysed further in this work).

Under the above assumptions and neglecting battery and transmission losses and communication issues, each community becomes equivalent to a single prosumer with aggregated PV and battery capacities over the full community. Let us consider as an example a particular grouping of 6 households $H = P \cup C$, including 1 battery-equipped prosumer p_1 among 3 prosumers $P = \{p_1, p_2, p_3\}$ and 3 consumers $C = \{c_1, c_2, c_3\}$, as the one on the left of Figure 1. Neglecting losses and communication faults, the community is then equivalent to a single larger entity with aggregated consumption from the 6 households, aggregated production from the 3 prosumers and having as much storage as p_1 . Suppose now the annual costs of all the households were 1000€ each for a total of 6000€. As a single community taking coordinated decisions, they may only have to pay 4800€, or 800€ each if the 1200€ gain is spread equally among participants. We use hereafter cost-saving for the reduction in cost obtained through cooperation, e.g. 1200€ in the previous example.

2.4 From Communities to Peer Matching

Among a pool (set of end-users willing to participate in P2P energy sharing), several communities can be managed independently from each other. How to partition efficiently a group of users into independent communities is the main focus of the present work. For any given partitioning, we can associate a global cost-saving corresponding to the sum of the cost-savings of each

community. Optimizing cost-efficiency of all resources in a given pool corresponds from a global perspective to maximizing the global cost-saving.

We assume customers can see and use their own data. but do not have access to their peers' data. To minimize data exchanges and reduce stress on the architecture, we assume the matching happens at a higher level through a third party dedicated entity. This centralized point of view assumes the end-users have subscribed to such an external service provider (being the energy provider or a third agent) in order to participate in the sharing process, and paying for the service through a fix share of the cost-saving obtained by each community. Under this setting, the service provider is in charge of grouping prosumers and consumers and supervising the transmission of data needed for the matching; its goal is thus to achieve the best global cost-saving to maximize its own benefit too. This also means that, once matching has been settled, communities can work in an independent fashion and do not need to rely on the service provider for managing their every day exchanges, neither for optimizing their electricity cost. Since endusers change consumption patterns and their will to participate through time, the matching can get updated or recomputed after a billing period has elapsed.

3 Algorithmic Modeling

We present in this section a formalism for the geographical peer matching problem in terms of finding a maximum-weight matching in bipartite hypergraphs. We then present how certain assumptions on the behaviors of the peers' preferences can simplify the problem.

3.1 Problem Abstraction

Preliminaries A hypergraph $\mathcal{G} = (V, E)$ is made of set of vertices V and a set of hyperedges $E \subseteq 2^V \setminus \emptyset$, where a hyperedge $e \in E$ is any non-empty subset of vertices. A hypergraph is said to be weighted if the hypergraph is associated with a weight function $w: E \to \mathbb{R}$. We say a hypergraph is k-bounded if all its hyperedges are of size at most k; a 3-bounded hypergraph is display in Figure 2(b).

A matching M in a hypergraph $\mathcal{G} = (V, E)$ is a set of disjoint hyperedges $(e_1 \text{ and } e_2 \text{ are considered disjoint})$ when they do not share any vertices, i.e., $e_1 \cap e_2 = \emptyset$). The weight of such a matching is the sum of the weights of the selected hyperedges that it contains, i.e., $w(M) = \sum_{e \in M} w(e)$, slightly abusing the notation. Given a partition of the vertices into two disjoint sets P and C, a bipartite hypergraph matching (BHM) is a matching of a hypergraph $G = (P \cup C, E)$ that contains in each selected hyperedges exactly one vertex within P, i.e., M is bipartite if $\forall e \in E$, $|e \cap P| = 1$. A k-bounded matching ensures that each output group is of size at most k; 3-bounded BHMs are displayed in Figure 2(d)-(f).

Geographical Peer Matching We define the Maximum-Weight Bipartite Hypergraph Matching (MWBHM) problem that consists in finding a bipartite hypergraph matching with maximum weight in a given hypergraph over vertex set $P \cup C$. The problem becomes dynamic when the weighting function w is time-dependent w_t and the matching problem should be solved for each time step. The Geographical Peer Matching (GPM) problem of parameter (k, Δ) is a bounded version of the MWBHM problem where we add the following three additional assumptions:

- 1. **Neighborhoods**: M is a k-bounded bipartite hypergraph matching, i.e., $\forall e \in M, |e| \leq k$.
- 2. **Spatiality**: peers are geographically distributed and the matching M should adhere to each peer's locality: every node of a selected hyperedge of M must be within geographical distance Δ of each other, i.e., $\forall e \in M, \forall v, v' \in e$, distance $(v, v') \leq \Delta$.
- 3. Computationally-intensive weights: weights are unknown beforehand but must be dynamically calculated; the function is considered expensive and the main bottleneck, hence the complexity of an algorithm solving the GPM problem is mainly measured in the number of weight computations.

In the following, we note $N_{\Delta}(v) = \{v' \mid \operatorname{distance}(v,v') \leq \Delta\}$ the neighborhood of vertex $v \in P$ and $\delta_{\Delta} = \frac{1}{|P|} \sum_{v \in P} |N_{\Delta}(v)|$ the average neighborhood size, used for complexity computations in the remaining. For instance, assuming that the nodes are spread uniformly and independently on a square zone of size $L \times L$ with toroidal properties to simplify¹, the average neighborhood size is $\delta_{\Delta} = (n-1) \cdot \pi(\Delta/L)^2 = \Omega(n)$ for $L = \Omega(\Delta)$.

Proposition 1. The GPM problem is NP-hard for $k \ge 3$, $\Delta > 0$.

Proof. In general, the hypergraph matching (HM) problem that consists in computing the maximum-weight matching of hyperedges is NP-hard, even for k-bounded hypergraphs for $k \geq 3$ [2, 37]. There is a trivial reduction to the weighted k-set packing problem, known to be NP-complete from Garey and Johnson [12]. Indeed, a hyperedge is nothing more than a subset of the vertex set, and the k-set packing problem [3] is looking for a maximum weight sub-collection of disjoint sets, which is equivalent to finding a maximum-weight matching of non-overlapping hyperegdes.

Now, adding the bipartite constraint is not reducing the difficulty of the problem, as one can reduce the GPM problem to HM as follows. Let $\mathcal{G} = (V, E)$ be a k-bounded hypergraph and let $\mathcal{G}' = (V_1 \cup V_2, E')$ be a hypergraph that contains all vertices of \mathcal{G} plus |E| "extra vertices", i.e., $V_1 = \{v_e \mid \forall e \in E\}$ and $V_2 = V$. Then all hyperedges of \mathcal{G}' are made of those of \mathcal{G} with

 $^{^1\}mathrm{A}$ torus topology means that each node has on average the same number of neighbors regardless of its position on the map and is very similar to the usual map when $L\gg\Delta.$

one extra vertex in each, that is $E' = \{e \cup \{v_e\} \mid e \in E\}$. Any matching M of E' is only made of edges that contain exactly one vertex of V_1 each, hence is a bipartite hypergraph matching under our definition. In turn, M is also a matching of \mathcal{G} (just discard the extra vertex in each hyperedge). At last, we can easily affect a position to each vertex so that all vertices are within distance Δ from each other for any $\Delta > 0$, hence defeating the additional spatial constraint.

When forgetting the spatial constraint, we note that for k=2, the problem becomes polynomial and is equivalent to finding the maximum-weight matching in a weighted bipartite graph, usually then named "the assignment problem". This is a classic problem where the Hungarian algorithm [21] provides the optimal solution in time $\mathcal{O}(nm+n^2\log n)$ for n vertices in the smaller vertex set and m edges, cf. [31]. Now adding spaciality and assuming δ_{Δ} as the average neighborhood size for search range Δ , this gives a running time of $\mathcal{O}(n^2\delta_{\Delta}+n^2\log n)=\mathcal{O}(n^3)$. We note that such a running time can be already prohibitive for large n and even for k=2, the problem becomes more challenging when considering its distributed equivalent [23].

Considering the difficulty of finding the optimal solution and the additional assumption on computational-intensive weights, an algorithm \mathcal{A} that produces a matching M as solution to the GPM should be evaluated on two main criteria: the **number of weights** that were computing by \mathcal{A} in order to find M, and the **quality of the solution** indicated by w(M). The rest of the section explains how the original P2P energy sharing setting translates into the above GPM abstraction. Refer to Table 1 for notation summary.

3.2 A Hypergraph Matching Problem

We can model the problem of forming P2P energy sharing communities in a continuous fashion as a (dynamic) GPM problem of parameters k and Δ . The underlying k-bounded hypergraph is $\mathcal{G} = (V, E)$ where the vertex set $V = P \cup C$ is made of P the set of prosumers (users equipped with renewable energy resources) and C the set of consumers (with no resources). The set of hyperedges E is made of all possible hyperedges that can be part of a k-bounded BHM (where k represents the maximum allowed size for the communities), i.e., $E \subset \{e \in P \times 2^C \mid 2 \le |e| \le k\}$. We explicitly remove communities made uniquely of consumers and singlenode communities (as those are completely useless in our context), as well as communities containing several prosumers (those generate very little gain), reducing the combinatorial possibilities for the hyperedges.

Table 1: Nomenclature used in the paper.

Symbol	Usage	Symbol	Usage
P	set of prosumers.	n	number of prosumers, i.e., $ P $.
C	set of consumers.	m	number of edges, i.e., $ P \cdot \bar{\delta_{\Delta}}$.
Δ	search radius.	k	maximum size for communities.
$\bar{\delta_\Delta}$	avg. neighborhood size.	w(e)	weight of the hyperedge e .

The weighting function w_t is dynamic and can be used to capture the cost-saving of a particular community at time t (forming the community is then equivalent to adding the corresponding hyperedge in the BHM). Limiting the search radius to Δ allows the service provider managing the P2P matching to use aggregators² in charge of smaller geographical areas. Remote control of the end-users' distributed resources could in turn be delegated to such aggregators relieving the users of data exchange and computational work during the P2P energy sharing process. In addition, having geographically closer communities allows to have better independence and load-balance in the system.

Since data is not known ahead of time, there are two strategies to compute the weight of a hyperedge: either using only past data, i.e., w_t is computed based on data recorded within some timespan $[t-\tau,t]$ for some τ , or using past and projected data. Since we consider in this work communities lasting for long period of time (months to years), the weight calculation based on projection is not the most appropriate solution as the accuracy of the projection degrade fast as the horizon grows (e.g. poor prediction is expected past 48h). In this context, peers affinity is more reasonably captured by setting τ to the same length as the billing period. To compute the weight of a hyperedge, necessary data must be transmitted (enduring some communication overhead) and a run of a LP-solver (requiring computation overhead) is required to solve the costoptimization problem (see § 2.2). The computation of a single weight is therefore both costly in terms of data exchanged and local computation, hence the matching must be computed while minimizing as much as possible computation of new weights.

3.3 A One-to-Many Assignment Problem

Let's assume that the weight of any group $H=p\cup\{c_1,\ldots,c_\ell\}$ with $\ell\leq k-1,\ p\in P$ and $c_i\in C$, can be calculated as the sum of the individual pairwise weights, i.e., $w(H)=\sum_{1\leq i\leq \ell}w(\{p,c_i\})$. In this situation, finding the best group (in terms of weight) of size k containing p is then equivalent of picking the k-1 best partners for p. Hence, the maximum weight matching can be reduced to a one-to-many assignment problem, that matches members of the set P with at most k-1 members of the set C such that the sum of the individual "edge weights" $w(\{p,c\})$ is maximum.

This problem can be further reduced to the classical and well known one-to-one assignment problem in the following manner: for each $p \in P$, make k-1 copies p^1, \ldots, p^{k-1} of node p, while keeping the original weights, i.e., $\forall c \in C, w(\{p^j, c\}) = w(\{p, c\})$. Finally solve the one-to-one assignment problem (maximum matching in bipartite graphs) with the input $P' = \{p^j \mid j \in [1..k-1], p \in P\}$ and C. As mentioned

²intermediate infrastructure level between end-users and service provider where data can be retrieved almost in real time and with fine granularity.

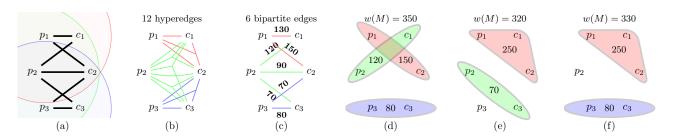


Figure 2: Illustration of the matching procedures with $P = \{p_1, p_2, p_3\}$, $C = \{c_1, c_2, c_3\}$ and k = 3: (a) E_{Δ} for a given search radius Δ , (b) all hyperedges based on E_{Δ} , (c) pairwise weights, and (d) Round Robin, (e) Single Pass, and (f) Classic Greedy matchings.

in § 3.1, the assignment problem can be solved exactly using the Hungarian algorithm in time $\mathcal{O}(|P'|^2 \cdot |C|) = \mathcal{O}(kn^2 \cdot \max\{\delta_{\Delta}, \log n\}) = \mathcal{O}(kn^3)$, which is already prohibitive for large systems (though some variations of the Hungarian algorithm may help, see e.g. [6]).

We note that setting the original weights (in the hypergraph) to the cost-saving do not follow this paradigm and even though finding the maximum matching of the pairs do provide a hypergraph matching, it is not guaranteed any longer to be maximal in terms of the sum of the weights of the hyperedges. Such an example is given in Figure 2(f) where $M = \{\{p_1, c_1, c_2\}, \{p_3, c_3\}\}$ maximizes $\sum_{\{p,c\} \in P \times C \mid \exists e \in M, \{p,c\} \in e} w(\{p,c\}) = 360$ but w(M) = 330 and thus M is not maximum, cf. the matching of Figure 2(d).

4 Peer Matching Algorithms

We describe in this section algorithms that produce a solution (i.e., a hypergraph matching) to the GPM problem (defined in § 3.1). The input of the matching problem is a bipartite graph based on prosumers location as defined in § 3.2. We first present three algorithms: Round Robin, Single Pass and Classic Greedy in § 4.1, then explain in § 4.2 how to instantiate them using different weight functions. We analyse in § 4.1.5 the time-complexity of the matching procedures; we note the weight function is constant in this analysis. Dealing with the latter, for each algorithm and weight function, we provide an asymptotic analysis of the number of time that weights are required to be computed in order to produce the solution in § 4.2. Note, to reduce the search space, we have restrained hyperedges to contain exactly one member of the set P (a prosumer in our context) but the algorithms can be easily adapted if this constraint is lifted.

4.1 Matching Algorithms

4.1.1 Definitions

Let $P \cup C$ the input of a GPM problem of parameter (k, Δ) . We assume that weight (p_i, c_j, M_i, t) returns a weight associated to node $p_i \in P$ and node $c_j \in C$ and possibly using a partially computed set of nodes $M_i \subset C$ that are already associated with p_i , whereas

t indicates the current time-step (weights are time-dependent). This function can be resolved by either:

- 1. executing a computationally-expensive procedure that relies on solving one or several LP-programs as defined in \S 2.2:
- 2. simply performing a look-up of a previously computed weight.

When a weight is obtained in the first case above, we say it is *computed*; such computation is the bottleneck of the matching procedures and is highlighted in the pseudocodes of the algorithms. The input of the matching algorithms is the bipartite graph $G = (P \cup C, E_{\Delta})$ where E_{Δ} captures all neighborhoods at geographic distance Δ from each prosumer, i.e., $E_{\Delta} = \{\{i,j\} \in$ $P \times C \mid \mathsf{distance}(p_i, c_j) \leq \Delta$; an example of E_Δ restraining possible hyperedges is illustrated in Figure 2(a)-(b). Note that the presented algorithms do not rely on the geographical positions of the nodes but on the authorised matching pairs captured by E_{Δ} , hence any bipartite graph can be given as input and the algorithms can be applied to a wider range of problems. The function order(P) sorts the set P according to a predefined ordering, to be provided by the user (cf § 5.1 for the ones used in the evaluation). As an example, the matchings obtained by the three algorithms are displayed in Figure 2(d)-(f) when using the pairwise weights of Figure 2(c) with processing order p_1, p_2, p_3 .

4.1.2 Round robin matching

Algorithm 1 builds a bipartite hypergraph matching by affecting to each prosumer one consumer at a time in a round-robin fashion. If no consumer can be affected to p (either because p has already k-1 consumers affected or no unmatched consumer is found within distance Δ of p), p is skipped (and discarded from subsequent matching attempts). Hence, all prosumers get at most 1 consumer each before a second iteration starts in affecting a second consumer to every prosumer. The greedy choice corresponds to selecting the consumer with highest weight; how the weights are settled is further described in § 4.2.

Algorithm 1: Round Robin Matching Procedure

```
Input: A bipartite graph G = (P \cup C, E_{\Delta}) and k \geq 2
    Output: M. a k-bounded bipartite hypergraph
                   matching;
    // Initialization
 1 foreach i \in P do
      M_i \leftarrow \emptyset;
    for each j \in C do
      S_j \leftarrow \text{False};
    \Psi \leftarrow \mathsf{order}(P)
5
    while \Psi \neq \emptyset do
 6
          for
each i \in \Psi do
7
                 N \leftarrow \{j \in C \mid \{i, j\} \in E_{\Delta} \land \neg S_j\} ;
 9
                if N = \emptyset \lor |M_i| = k - 1 then
                      \Psi \leftarrow \Psi \setminus \{i\} ;
10
                \mathbf{else}
11
                      if |N| > 1 then
12
                             for
each j \in N do
13
                                    b_j \leftarrow \mathsf{weight}_t(p_i, c_j, M_i);
15
                             \ell \leftarrow \arg\max_{j \in N} b_j ;
16
                       else
                           \ell \leftarrow N[1]
17
                       S_{\ell} \leftarrow \text{True};
18
                      M_i \leftarrow M_i \cup \{\ell\};
19
20 return \{M_i \mid i \in P\};
```

Algorithm 2: Single-Pass Matching Procedure

```
Input/Output: as in Algorithm 1.
    // Same as lines 1-5 in Algorithm 1
 1 foreach i \in P do
 2
          N \leftarrow \{j \in C \mid \{i, j\} \in E_{\Delta} \land \neg S_j\} ;
         if |N| \ge k then
 3
               for
each j \in N do
 4
                      b_j \leftarrow \mathsf{weight}_t(p_i, c_j, M_i);
 5
 6
               while |M_i| < k-1 do
                    \ell \leftarrow \arg\max_{j \in N} b_j;
 7
                    M_i \leftarrow M_i \cup \{\ell\};
 8
 9
          else
               M_i \leftarrow N;
10
         foreach j \in M_i do
11
              S_i \leftarrow \text{True};
13 return \{M_i \mid i \in P\};
```

4.1.3 Single pass matching

Contrary to the previous one, Algorithm 2 builds a matching in a single pass over the prosumers. For each prosumer p, the k-1 best available consumers (in terms of weight) are matched with p.

4.1.4 Classic greedy matching

Algorithm 3 is the "classic" greedy procedure for solving the assignment problem (cf. § 3.3) that, based on pre-computing all pairwise weights, sorts the prosumer-consumer pairs $(p,c) \in E_{\Delta}$ from highest to lowest, then associates consumers to prosumer whenever possible (c not already matched and p having less than k-1 affected consumers).

4.1.5 Time complexity

Proposition 2. Algorithm 1, 2 and 3 3 run in respectively $\mathcal{O}(mk)$, $\mathcal{O}(m)$ and $\mathcal{O}(m\log m)$ time.

Proof. Algo. 1: the main while-loop is executed at most k times (after what all prosumers have been removed by line 10) and each inner for-loop goes through |P| prosumers, each time calculating the best local choice using $\mathcal{O}(|N_{\Delta}(p)|)$ for prosumer p. The inner for-loop thus takes total time $\mathcal{O}(|P| + |E_{\Delta}|) = \mathcal{O}(\bar{\delta_{\Delta}}|P|)$.

Algo. 2: line 5 is executed $\mathcal{O}(|E_{\Delta}|)$ times whereas lines 6-8 also take $\mathcal{O}(m)$ in total when using a selection and partition algorithm to find the k-1 highest unsorted weights in each neighborhood.

Algo. 3: this is the time needed to sort all weights, then going through the sorted list takes $\mathcal{O}(|P|\bar{\delta_{\Delta}})$ time.

4.2 Instantiate the Weight Function

We define here two ways to calculate the weights: either in a "memoryfull" fashion, taking into account previous choices made by the matching algorithm, or in a "memoryless" one with constant weights only depending on the involved prosumer and consumer pair (p_i, c_i) being examined. For each variant, we propose two ways to calculate the weight for a given pair, either based on the energy cost produced by the pair working as a community in the recent past $[t - \tau, t]$ (running a single LP-solver as described in § 2.2 and 2.3) or based on the cost-saving produced by the pair over the same period $[t-\tau,t]$. In the former case, the goal of the GPM problem being to minimize the total cost, we inverse the weights to keep a maximum-weight problem. In all, four weight functions are defined as $\mathbf{WX}_t(i,j)$ to instantiate the call to weight, (p_i, c_i, M_i) .

4.2.1 Memoryless weights

Using memoryless weights (WA and WB) means that the weights are constant and independent of the matching procedure being run. This also means that building the matching is nothing more than solving the classic assignment problem (as described in § 3.3). An example of memoryless weights is given in Figure 2(c) where a weight is given for each pair $(p,c) \in P \times C$. We set the memoryless weights as follows:

```
WA_t(i, j) = -bill(\{p_i, c_i\}, [t - \tau, t]);
```

Algorithm 3: Classic Greedy Matching Procedure

```
Input/Output: as in Algorithm 1.

// Same as lines 1-5 in Algorithm 1

1 foreach \{i,j\} \in E_{\Delta} do

2 | b_{i,j} \leftarrow \text{weight}_t(p_i,c_j,M_i);

// Sort all possible matching pairs by decreasing weights

3 B \leftarrow \text{decreasing\_sort}(\{b_{i,j} \mid \{i,j\} \in E_{\Delta}\});

4 foreach b_j \in B do

5 | if |M_i| < k - 1 \land \neg S_j then

6 | S_j \leftarrow \text{True};

7 | M_i \leftarrow M_i \cup \{j\};

8 return \{M_i \mid i \in P\};
```

$$\mathbf{WB}_t(i,j) = \sum_{x \in \{p_i, c_j\}} \mathbf{bill}(\{x\}, [t - \tau, t])) + \mathbf{WA}_t(i,j).$$

Proposition 3. Algorithms 1, 2 and 3 need to compute $\mathcal{O}(m)$ memoryless weights to solve the (k, Δ) -GPM problem.

Proof. Note Algorithms 2 and 3 only call the weight routine once per prosumer-consumer pair, hence $\mathcal{O}(|E_{\Delta}|)$ weights are ever calculated. By remembering previously computed weights, Algorithm 1 needs also to compute each weight only once since the current matching is not involved when using memoryless weights. \square

Regarding the minimum number of weights to compute, the three algorithms differ. Algorithm 3 needs to sort all the weights, so $|E_{\Delta}|$ weights are always computed regardless how the nodes are distributed in space. However, for Algorithm 1 and Algorithm 2, there exist cases where none of $\Omega(|P|^2)$ weights are computed³.

It is possible to pre-compute all the $\mathcal{O}(n^2)$ memoryless weights based on some E_{Δ} , however, the necessary number of weights to compute may be lower when using Algorithm 1 and Algorithm 2.

4.2.2 Memoryfull weights

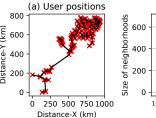
We define two memoryfull weight functions (WC and WD) to take into account the already matched pairs, hence trying to provide the most accurate answer to the bipartite hypergraph matching problem. The weights are set as follows:

$$\begin{aligned} \mathbf{WC}_{t}(i,j) &= -\mathbf{bill}(M_{i} \cup \{p_{i},c_{j}\},[t-\tau,t]); \\ \mathbf{WD}_{t}(i,j) &= \sum_{x \in M_{i} \cup \{p_{i},c_{j}\}} \mathbf{bill}(\{x\},[t-\tau,t])) + \\ \mathbf{WC}_{t}(i,j). \end{aligned}$$

Proposition 4. Algorithms 2 and 3 need to compute $\mathcal{O}(m)$ weights when using memoryfull weights for solving the GPM problem of parameter (k, Δ) ; Algorithm 1 needs to compute $\mathcal{O}(km)$ weights.

Proof. Variants C and D now depend on previous matching. As with memoryless weights, Algorithms 2 and 3 only call the weight routine at most once per edge in E_{Δ} , hence $\mathcal{O}(|E_{\Delta}|)$ weights are ever used. For Algorithm 1, the number of weight calculation is bounded by its time complexity, cf. proposition 2.

For Algorithm 1, pre-computation is not needed as each time the weight function is called, a weight calculation must occur (i.e., M_i is different for each iteration). Note that all the algorithms compute significantly fewer weights than the number of hyperedges, that is bounded by $\mathcal{O}\left(|P|\cdot\binom{\delta}{k-1}\right)=\mathcal{O}(n\delta^{k-1})=\mathcal{O}(n^k)$, with δ being the maximum size of the neighborhoods, i.e., $\delta=\max_{p\in P}|N_{\Delta}(p)|$.



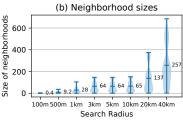


Figure 3: (a) Position of the users with an example of grid infrastructure relying them; (b) Size of the neighborhoods.

4.2.3 Sampling for Reducing the Search Space

One solution to reduce the number of weights to compute is to sample the neighborhoods, before executing the matching procedure. We propose here two sampling methods for a given sampling size s. For neighborhoods containing less than s consumers, keep them all, otherwise for neighborhood N:

- Random: pick uniformly at random s neighbors in N;
- **Greedy**: pick the s consumers with largest consumptions in N (expected to produce higher cost-savings than the others).

Applying such a sampling step as a pre-processing of the neighborhoods reduces the number of weights to compute to only $\mathcal{O}(ns)$.

5 Experimental Study

We present in this section an experimental study comparing the performance of the proposed algorithms in terms of number of computed weights (cf. § 4.1.1) versus the quality of the output solution (amount of cost-saving produced by the computed matching).

5.1 Experimental Set-up

Energy Data Consumption profiles of 2221 real households are used in this study, originating from [28]. Each trace contains electricity consumption measures on a hourly basis for a different household for one year. Households are equipped with energy resources as follows: 10% of them are prosumers with only PV panels, 10% are prosumers with both PV panels and a battery, the rest are usual consumers. Capacities for energy resources are set for each prosumer relative to its average electricity consumption following current installed PV capacities (based on the Open PV 19 dataset [27]). A hourly solar profile is used here assuming same roof panel orientations [26]. We use in our study the original city position of the households (obtained through using their respective zipcode), where we added a small random offset of up to 1km, see Figure 3(a). The hourly electricity price profile is based on the output

 $^{^3 \}text{One can indeed construct the following example (for Algorithm 1): } p_i \text{ is positioned at } (0, i \cdot \varepsilon) \text{ and } c_j \text{ is positioned at } (0, \Delta + j \cdot \varepsilon') \text{ with } 0 < \varepsilon < \frac{\Delta}{|P|} \text{ and } \varepsilon < \varepsilon' < \frac{\varepsilon(1+|C|)}{|C|}.$

from a European scale dispatch model [15]. Taxes are set to $\mathbf{tax} = 25\%, \mathbf{el}_{tax} = 6.9 \in \text{cents/kWh}$ and $\mathbf{el}_{net} = 0.58 \in \text{cents/kWh}$.

Algorithms We compare in this study the three procedures defined in § 4.1: Round Robin, Single Pass and Classic Greedy algorithms. For the first two, the order in which prosumers are processed can take three form: Increasing (Incr.) or Decreasing (Decr.) order of their average consumptions, or decreasing order of their energy resources (Rsc.); the latter is obtained by adding both PV capacity (kWp) and battery capacity (kWh). Single Pass and Classic Greedy algorithms can be tuned to use two variants of the weight function (as explained in § 4.2): cost-based weights (WA) or saving-based weights (WB). In addition to the above two variants (referred as "memoryless weights"), the Round Robin algorithm is tested with the two memoryfull counterparts: cost-based memoryfull weights (WC) or saving-based memoryfull weights (WD). We tested 13 different combinations of algorithm, prosumer order and weight function (cf. first three columns of Table 2). The experiments were run on a high-end server (Intel Xeon E5 2650 CPU, 64GB RAM) where computing a single weight takes about 2sec.

Search radius We set for the experiments 8 different search radii: 100m, 500m, 1km, 3km, 5km, 10km, 20km and 40km. For each search radius, Figure 3(b) presents the distribution of the size of the nodes' neighborhoods (only based on the node positions). Since the search radii of 5km and 10km produce very similar neighborhood sizes as 3km, we omit them in the rest of the evaluation.

5.2 Comparisons of Peer Matching Algorithms

We shall hereafter compare the performance of the introduced algorithms based on our two performance measures: quality of the matching (in terms of costsaving achieved by the communities) and number of weights to compute (main computational bottleneck).

5.2.1 Quality of the solution

The solution obtained using the studied algorithms is summarized in relative terms in Table 2. The results are calculated upon matching together 445 prosumers (half of them having also a battery system) with 1776 consumers into groups of size at most 5 containing exactly one prosumer each and running the LP-solver (cf. § 2.2) over a year of data to obtain each community's yearly cost (used as the hyperedge weight). Up to $150000 \in$ can be saved when all the nodes cooperate in a single community (equivalent of setting $\Delta = k = \infty$), from which up to 90% can be recovered by using $\Delta = 40 \text{km}$ and k = 5. In terms of quality of the solution, the single pass algorithm using decreasing order and cost-based weights is clearly outperformed by the rest, whereas

Table 2: Comparison of the different algorithms, weight functions and processing orders \star .

Algo.	Order	WF	100m	500m	1km	3km	20km	40km
Round Robin	Incr.	WA	9.5%	60.9%	74.2%	78.8%	81.6%	83.8%
		WB	9.3%	61.7%	73.7%	78.5%	80.3%	82.4%
		WC	9.5%	60.9%	74.3%	78.9%	81.6%	83.9%
		WD	9.3%	61.7%	74.1%	79.0%	81.0%	83.2%
	Decr.	WA	9.5%	61.7%	74.8%	79.0%	82.2%	85.0%
		WB	9.6%	64.4%	77.9%	82.9%	85.0%	87.4%
		WC	9.5%	74.7%	61.7%	79.0%	82.1%	84.8%
		WD	9.6%	64.4%	78.1 %	83.4%	85.6%	88.3%
	Rsc.	WB	9.6%	58.7%	77.9%	82.8%	84.8%	87.3%
Single Pass		WB	9.5%	61.1%	75.0%	79.7%	83.6%	86.9%
	Decr.	WA	9.5%	58.7%	68.8%	71.1%	74.7%	77.6%
		WB	9.5%	61.1%	73.6%	78.7%	82.5%	86.0%
Classic Greedy		WB	9.6%	64.3%	77.5%	83.2%	86.7%	90.5%

^{*} the percentages are the fraction of the single 2221-households obtained by the matchings; best results for each radius is in bold and green background.

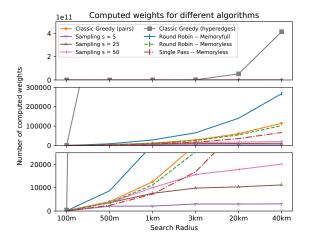


Figure 4: Average number of calculated weights for the different matching algorithms.

the two best performing ones are the classic greedy and the round robin algorithm using decreasing order (both saving-base weights). Closely behind, we note the high performance of both the round robin (decreasing order, WB) and the single pass (WB) algorithms, both requiring significantly less weights to compute.

5.2.2 Number of weights to compute

Figure 4 presents the number of weights⁴ calculated along the computation of the matchings of Table 2. All algorithms (except the sampling ones) display a quadratic increase in number of weights as the search radius increases, with varying slopes. Confirming the theoretical upper bounds given in Section 4, memoryfull weights entail a large computational overhead. On the contrary, memoryless weights reduce the burden on cal-

⁴Variability is negligible: the average value is displayed and error bars, if any, indicate min-max accounting the different tested weight functions and prosumer orders.



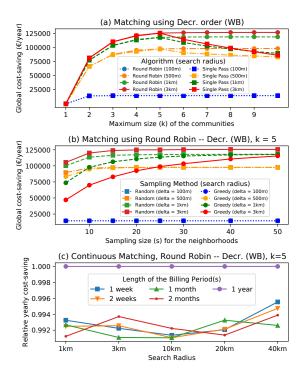


Figure 5: Sensitivity analysis showing the impact of (a) different maximum neighborhood sizes, (b) presampling the neighborhoods and (c) shortening the billing period.

culating weights with the single pass algorithm only calculating about 57% of all prosumer-consumer pairs, and less than 25% of the memoryfull weights. As expected, sampling s values among the neighborhoods further reduces the weight computation to a bit less than ns in total. All algorithms compute several orders of magnitude less weights than the number of possible communities, at about 414 billion weights for $\Delta = 40km$, that would be required to compute in order to run a classic greedy algorithm on the possible hyperedges (instead of on the prosumer-consumer pairs).

Neighborhood size 5.2.3

The neighborhood size providing the best trade-off depends on the search radius, as illustrated by Figure 5(a) showing the cost-saving produced using different bounds on the community size (k). In this scenario, using k = 5 provides a cost only 0.01% inferior than using k = 10 for the round robin algorithm. For the single pass algorithm, using neighborhood sizes above 5 is detrimental as the first prosumers get matched with a large amount of peers which removes the possibility for later processed prosumers to get consumers at all. We note choosing k should follow the distribution of prosumers in the pool.

5.2.4Weights sampling

We analyse the effect of weight sampling in Figure 5(b). The two strategies to sample a fixed number of po-

tential partners in each neighborhood are compared: either perform a random or a greedy selection. The result clearly favors a random selection which is very efficient even for small sampling size (here, 2k to 4k), whereas the greedy solution only catches up for large sampling size (both converges to the non-sampling algorithm when s reaches δ , the maximum size of the neighborhoods).

5.2.5Continuous matching

Impact of performing the peer matching on smaller intervals than one year is studied in Figure 5(c). The figure displays in relative term the cost-saving achieve when performing the peer matching every week, second week, month, and second month, with all compared to the one year matching used so far (set to 1). The costsavings obtained with all smaller periods decrease compare to using a year of data, but only to a very small extend, all reaching at least 99% of the yearly saving. Even though shorter interval communities could gain from seasonable changes in finding ideal trading partner among fellow peers, this confirms that long-term communities are robust in terms of cost-saving.

5.3Summary of the Results

In short, our experimental evaluation advocates that:

- 1. to maximize cost-efficiency, use Round Robin (decreasing order, WD) for a search radius up to 3km and Classic Greedy beyond;
- 2. memoryfull weights do not provide a significant advantage in terms of cost-saving that justifies the induced overhead, with Round Robin (decreasing or resource order, WB) algorithm providing the best trade-off overall;
- 3. applying a random pre-sampling of each neighborhood of about s = 20 weights for k = 5 reduces drastically the computational overhead while having little impact on the cost-efficiency, with further reductions obtained using the Single Pass algorithm.

Related Work 6

P2P energy sharing has been in the focus of numerous research works in recent years (cf. the comprehensive survey [34] and references therein). Among those works, we can distinguish two types of P2P sharing communities. Short-term coalitions (lasting for e.g. 10min) are formed to cover a single timeslot where game-theoretic approaches are used to optimize the individual gains among other things [33]. On the contrary, long-term coalitions [22, 18, 4, 11] seek to form communities that will take coordinated decisions within the same group of peers over months to years. Often also associated with geographical closeness, the long-term communities present many advantages from an infrastructure

point of view, being able to better regulate local loadbalancing of distributed energy resources, providing higher local self-sufficiency and/or reducing the local peak demand. Research on optimizing the gain outcome from the peering process using constrained optimization has focused on different aspects: how communications are handled [22], reaching stable partitions [5, 4], finding optimal resources based on community sizes [18], privacy aspects and amount of data transmitted over the network [20, 10, 11], etc. Small neighborhoods have been shown to provide a high share of possible gain while being favorable in terms of data exchange [11] and different matching were previously studied in [5, 4], based upon stable partitions (i.e., nodes with self-interest) and on a cost-sharing mechanism known beforehand. However, the algorithmic problem behind the formation of localized and long-term P2P energy sharing communities with a global objective has not been studied before to the best of our knowledge, neither analytically nor from a practical point of view.

The solutions explored in this paper relate to the hypergraph matching problem, especially to the works proposing mechanisms to compute practical solutions. All the best heuristic algorithms that have been proposed to solve the problem are based on the notion of "local search" [1]. Local search consists in incrementally improving a starting solution by performing a series of small changes (typically switching membership of a node from one partition to another). The most competitive algorithm for the hypergraph matching problem is given by Berman [2], providing a (k+1)/2approximation algorithm. The algorithm is based on finding independent set in d-claw free graphs, and it requires a series of prepossessing steps in order to solve the hypergraph matching problem, e.g. [37] makes these steps explicit to obtain a hypergraph matching algorithm from Berman's algorithm. MWHM has also been studied in a distributed setting [7], as well as under "bmatching" generalizations [29] where nodes can appear in b different hyperedges instead of one only.

All local search-based algorithms, however, assume access to the full list of weights since their starting point is the output of what we referred here as the "classic greedy" algorithm. Also, those algorithms have a time complexity that is exponential in k [3]. In [24], computing all the weights was already identified as a computational bottleneck when the number of possible hyperedges is large, proportional to $\mathcal{O}(n^k)$ when there are n nodes in the input hypergraph. The authors hence propose a heuristic algorithm specific to their problem that only necessitates to compute $\mathcal{O}(kn^2)$ weights using an algorithm of time-complexity $\mathcal{O}(kn^3)$. For the number of computed weights, this matches the same bound shown here when using memoryfull weights and we further reduce it to only $\mathcal{O}(n^2)$ weights while introducing the notion of "memoryless weights". All the algorithms studied in this work have also noticeable smaller time complexities, namely $\mathcal{O}(kn^2)$, $\mathcal{O}(n^2)$ and $\mathcal{O}(n^2 \log n)$.

Note the problem studied here adds three additional constraints to hypergraph matching: requirement for a bipartite solution, spacial constraints and a challenging environment where the weight of a hyperedge is computationally expensive to obtain.

7 Conclusion

This paper studies the peer matching problem to participate in P2P energy sharing. We introduce the Geographical Peer Matching problem within a well-defined mathematical framework setting the problem as a hypergraph matching problem with a bounded search radius Δ and output partitions of size up to k. This allows known approximation algorithms for the weighted k-set problem to be ported to our energy sharing setting. To provide an efficient solution to the problem, we introduce and analyze three different matching algorithms that do not require to compute all $\mathcal{O}(n^k)$ possible weights (each requiring the run of a LP-solver) but only at most $\mathcal{O}(kn^2)$. The introduced algorithms are both scalable in terms of required computation and efficient in terms of the quality of the computed solution. Our extensive experimental study indeed shows that up to 90% of the benefit of an unrealistic unbounded matching (i.e., setting k = n and $\Delta = \infty$) can be obtained by limiting communities to 5 nodes only with $\Delta = 40 \mathrm{km}$, while 84% can already be reached with communities of diameter $\Delta = 3$ km. We also provide optimizations that, even though do not change asymptotic behaviors of the proposed algorithms, are shown in this work to yield a practical computational advantage without sacrificing the quality of the produced solution. We expect that as the introduced algorithms are more general than our specific problem, they can also be useful in other contexts.

Some practical aspects for further research are (1) how to push parts of the matching computation towards the end-users for a more edge-friendly solution, for saving data transfers and caring about privacy perspective [9, 17], while transitioning from batch-based to the online analysis required by today's smart metering infrastructure [36, 35], and (2) how to update the matching dynamically and maintain a stable network through the arrivals and departures of peers [8, 14], possibly building on and adapting previous distributed and adaptive algorithms for matching with preferences [13, 19].

References

- [1] Esther M Arkin and Refael Hassin. On local search for weighted k-set packing. *Mathematics of Operations Research*, 23(3):640–648, 1998.
- [2] Piotr Berman. A d/2 approximation for maximum weight independent set in d-claw free graphs. In *Scandinavian Workshop on Algorithm Theory*, pages 214–219. Springer, 2000.
- [3] Barun Chandra and Magnús M Halldórsson. Greedy local improvement and weighted set

- packing approximation. Journal of Algorithms, 39(2):223–240, 2001.
- [4] Sid Chi-Kin Chau, Khaled Elbassioni, and Yue Zhou. Approximately socially-optimal decentralized coalition formation. arXiv preprint arXiv:2009.08632, 2020.
- [5] Sid Chi-Kin Chau, Jiajia Xu, Wilson Bow, and Khaled Elbassioni. Peer-to-peer energy sharing: Effective cost-sharing mechanisms and social efficiency. In *Proceedings of the 10th ACM Int'l Conf.* on Future Energy Systems, pages 215–225. ACM, 2019.
- [6] Hong Cui, Jingjing Zhang, Chunfeng Cui, and Qinyu Chen. Solving large-scale assignment problems by kuhn-munkres algorithm. In 2nd Int. Conf. Advances Mech. Eng. Ind. Inform. (AMEII 2016), 2016.
- [7] Tao Cui, Lijun Chen, and Tracey Ho. Distributed optimization in wireless networks using broadcast advantage. In 2007 46th IEEE Conf. on Decision and Control, pages 5839–5844. IEEE, 2007.
- [8] Philippe Duchon and Romaric Duvignau. Local update algorithms for random graphs. In *Latin American Symposium on Theoretical Informatics*, pages 367–378. Springer, 2014.
- [9] Romaric Duvignau, Vincenzo Gulisano, Marina Papatriantafilou, and Vladimir Savic. Streaming piecewise linear approximation for efficient data management in edge computing. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pages 593–596, 2019.
- [10] Romaric Duvignau, Verena Heinisch, Lisa Göransson, Vincenzo Gulisano, and Marina Papatriantafilou. Small-scale communities are sufficient for cost-and data-efficient peer-to-peer energy sharing. In Proceedings of the Eleventh ACM Int'l Conf. on Future Energy Systems, pages 35–46, 2020.
- [11] Romaric Duvignau, Verena Heinisch, Lisa Göransson, Vincenzo Gulisano, and Marina Papatriantafilou. Benefits of small-size communities for continuous cost-optimization in peer-to-peer energy sharing. Applied Energy, 301:117402, 2021.
- [12] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.
- [13] Giorgos Georgiadis and Marina Papatriantafilou. Adaptive distributed b-matching in overlays with preferences. In *Int'l Symposium on Experimental Algorithms*, pages 208–223. Springer, 2012.
- [14] Giorgos Georgiadis and Marina Papatriantafilou. Overlays with preferences: Distributed, adaptive approximation algorithms for matching with preference lists. *Algorithms*, 6(4):824–856, 2013.

- [15] Lisa Göransson, Joel Goop, Thomas Unger, Mikael Odenberger, and Filip Johnsson. Linkages between demand-side management and congestion in the european electricity transmission system. *Energy*, 69:860–872, 2014.
- [16] Ulf JJ Hahnel, Mario Herberz, Alejandro Pena-Bello, David Parra, and Tobias Brosch. Becoming prosumer: Revealing trading preferences and decision-making strategies in peer-to-peer energy communities. *Energy Policy*, page 111098, 2019.
- [17] Bastian Havers, Romaric Duvignau, Hannaneh Najdataei, Vincenzo Gulisano, Marina Papatriantafilou, and Ashok Chaitanya Koppisetty. Driven: A framework for efficient data retrieval and clustering in vehicular networks. Future Generation Computer Systems, 107:1–17, 2020.
- [18] Verena Heinisch, Mikael Odenberger, Lisa Göransson, and Filip Johnsson. Organizing prosumers into electricity trading communities: Costs to attain electricity transfer limitations and self-sufficiency goals. *Int'l Journal of Energy Research*, 2019.
- [19] Arif Khan, Alex Pothen, Md Mostofa Ali Patwary, Nadathur Rajagopalan Satish, Narayanan Sundaram, Fredrik Manne, Mahantesh Halappanavar, and Pradeep Dubey. Efficient approximation algorithms for weighted b-matching. SIAM Journal on Scientific Computing, 38(5):S593-S619, 2016.
- [20] Ali Khodabakhsh, Jimmy Horn, Evdokia Nikolova, and Emmanouil Pountourakis. Prosumer pricing, incentives and fairness. In *Proceedings of the 10th* ACM Int'l Conf. on Future Energy Systems, pages 116–120. ACM, 2019.
- [21] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.
- [22] Chao Long, Jianzhong Wu, Yue Zhou, and Nick Jenkins. Peer-to-peer energy sharing through a two-stage aggregated battery control in a community microgrid. *Applied energy*, 226:261–276, 2018.
- [23] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. Journal of the ACM (JACM), 62(5):1–17, 2015.
- [24] Xiaofu Ma, Qinghai Gao, Vuk Marojevic, and Jeffrey H Reed. Hypergraph matching for mu-mimo user grouping in wireless lans. Ad Hoc Networks, 48:29–37, 2016.
- [25] Fredrik Manne, Md Naim, Håkon Lerring, and Mahantesh Halappanavar. On stable marriages and greedy matchings. In 2016 Proceedings of the 7th SIAM Workshop on Combinatorial Scientific Computing, pages 92–101. SIAM, 2016.

- [26] Zack Norwood, Emil Nyholm, Todd Otanicar, and Filip Johnsson. A geospatial comparison of distributed solar heat and power in europe and the us. *PloS one*, 9(12):e112442, 2014.
- [27] NREL. The open pv project. National Renewable Energy Laboratory, 2019.
- [28] Emil Nyholm, Joel Goop, Mikael Odenberger, and Filip Johnsson. Solar photovoltaic-battery systems in swedish households – self-consumption and selfsufficiency. *Applied energy*, 183:148–159, 2016.
- [29] Ojas Parekh and David Pritchard. Generalized hypergraph matching via iterated packing and local ratio. In *Int'l Workshop on Approximation and Online Algorithms*, pages 207–223. Springer, 2014.
- [30] Amrit Paudel, Kalpesh Chaudhari, Chao Long, and Hoay Beng Gooi. Peer-to-peer energy trading in a prosumer-based community microgrid: A game-theoretic model. *IEEE Trans. on Industrial Electronics*, 66(8):6087–6097, 2018.
- [31] Lyle Ramshaw and Robert E Tarjan. On minimum-cost assignments in unbalanced bipartite graphs. HP Labs, Palo Alto, CA, USA, Tech. Rep. HPL-2012-40R1, 2012.
- [32] Ruggero Schleicher-Tappeser. How renewables will change electricity markets in the next five years. *Energy policy*, 48:64–75, 2012.
- [33] Wayes Tushar, Tapan Kumar Saha, Chau Yuen, Thomas Morstyn, H Vincent Poor, Richard Bean, et al. Grid influenced peer-to-peer energy trading. *IEEE Trans. on Smart Grid*, 11(2):1407–1418, 2019.
- [34] Wayes Tushar, Chau Yuen, Tapan K Saha, Thomas Morstyn, Archie C Chapman, M Jan E Alam, Sarmad Hanif, and H Vincent Poor. Peer-topeer energy systems for connected communities: A review of recent advances and emerging challenges. Applied Energy, 282:116131, 2021.
- [35] Joris van Rooij, Vincenzo Gulisano, and Marina Papatriantafilou. Locovolt: Distributed detection of broken meters in smart grids through stream processing. In *Proceedings of the 12th ACM Int'l Conf. on Distributed and Event-based Systems*, pages 171–182, 2018.
- [36] Joris van Rooij, Johan Swetzén, Vincenzo Gulisano, Magnus Almgren, and Marina Papatriantafilou. echidna: Continuous data validation in advanced metering infrastructures. In 2018 IEEE Int'l Energy Conf. (ENERGYCON), pages 1–6. IEEE, 2018.
- [37] Long Zhang, Hongliang Zhang, Lisu Yu, Haitao Xu, Lingyang Song, and Zhu Han. Virtual resource allocation for mobile edge computing: A hypergraph matching approach. In 2019 IEEE Global

- Communications Conf. (GLOBECOM), pages 1–6. IEEE, 2019.
- [38] Zhenyuan Zhang, Haoyue Tang, Qi Huang, and Wei-Jen Lee. Two-stages bidding strategies for residential microgrids based peer-to-peer energy trading. In 2019 IEEE/IAS 55th Industrial and Commercial Power Systems Technical Conf. (I&CPS), pages 1–9. IEEE, 2019.
- [39] Yue Zhou and Sid Chi-Kin Chau. Multi-user coalition formation for peer-to-peer energy sharing. In *Proceedings of the Eleventh ACM Int'l Conf. on Future Energy Systems*, pages 386–387, 2020.