A Game Theoretic Approach to the Formation of Clustered Overlay Networks (Supplemental Material)

The material in this addendum is structured as follows. In Section 1, we define an alternative measure to social cost for measuring the overall quality of the clustered overlay. Section 2 presents the derivation of the case study results, while Section 3 describes the coordinated protocol in detail. Section 4 presents additional experimental results and Section 5 additional related research. Finally, the Appendix includes all proofs of the Lemmas and Propositions in [14].

1 Alternative Cost Model

The social cost defines the quality of a configuration based on the individual costs of all nodes, i.e., players. We can also evaluate the overall quality or cost of a configuration from a query workload perspective. In this respect, the recall cost for a cluster configuration is the average recall cost for attaining results for all queries in Q.

Definition 1 (Workload Cost): The workload cost of a cluster configuration S is:

$$WCost(S) = \alpha \sum_{c_k \in C} \frac{|c_k|\theta(|c_k|)}{|V|} + \sum_{q \text{ in } Q} \frac{num(q,Q)}{num(Q)} \sum_{n_i \text{ s.t. } q \text{ in } Q(n_i)} \frac{num(q,Q(n_i))}{num(q,Q)} \sum_{n_j \notin V(s_i)} r(q,n_j)$$

The first term models the cost for maintaining the clusters, while the second one models the recall loss for all queries, i.e., the cost for evaluating them outside the clusters of their initiator.

The main difference between the social and the workload cost lies on how they assign weights to queries. In the social cost, the weight or importance of each query is based on its frequency in the local workload of each node, whereas, in the workload cost the weight of each query is based on its frequency in the global query workload. Intuitively, while the social cost regards all nodes as equals, the workload cost considers more demanding nodes, i.e. nodes that pose more queries, as more important than low demanding ones. The following proposition relates the two costs for equally demanding nodes.

Proposition 1: If for all nodes $n_i, n_j \in V$, $num(Q(n_i)) = num(Q(n_j)) = \frac{num(Q)}{|V|}$, then improving the social cost improves the workload cost and vice versa.

Proof. Using the definition of individual cost (Def. 1 in [14]), the social cost can be written as:

$$SCost(S) = \alpha \sum_{n_i \in V} \sum_{c_k \in s_i} \frac{\theta(|c_k|)}{|V|} + \sum_{n_i \in V} \sum_{q \text{ in } Q(n_i)} \frac{num(q,Q(n_i))}{num(Q(n_i))} \sum_{n_j \notin V(s_i)} r(q,n_j)$$

The membership cost of $SCost$ is equal to the first term of $WCost$. Just consider that each cluster c_k appears in the sum of $SCost$ as many times as the nodes that belong to it, i.e., its size $|c_k|$. The second term differs from the second term of $SCost$ only on how much the workload of each node is taken into account. It is easy to see, that if nodes get an equal part of the query workload, i.e., $num(Q(n_i))$ = $num(Q(n_j))$, for all nodes n_i, $n_j \in V$, the recall parts of the two costs are proportional. Thus, we deduce that improving one of the costs, also improves the other.□

2 Derivation of Case Study Results

Next, we provide the derivation of the results summarized in Table 1 in [14].

In general, our game is an non-cooperative asymmetric game. A game is *asymmetric*, if the values of the utility function or payoff differ, if different players select the same strategy. However, there are cases in which the content and query distribution among the nodes is such that the game is symmetric. For instance, the case where there is no underlying clustering is a symmetric one. For symmetric games, the following holds.

Observation 1: Let $n_i \in V$ be a node and s_i its optimal strategy, that is, the strategy that minimizes the value of its individual cost. If the game is symmetric for the nodes in V, then s_i is also optimal for all nodes in V.

Next, we define formally the three characteristic cases presented in [14].

CASE I: NO UNDERLYING CLUSTERING. All nodes in V are considered similar in the following sense:

$$\forall n_i, n_j \in V,\ num(Q(n_i)) = num(Q(n_j)) = num(Q)/|V| \text{ and } \forall\ q \text{ in } Q,\ r(q, n_i) = r(q, n_j) = 1/|V|.$$

Note that in this case, our game becomes symmetric, since all players yield the same payoffs when applying the same strategy.

CASE II: SYMMETRIC SCENARIO. There are g ($g \geq 1$) disjoint groups of nodes of the same size ($|V|/g$) such that the members of each group offer and request content from the same category.

$$\forall\ n_i\ n_j \text{ in the same group, } num(Q(n_i)) = num(Q(n_j)) \text{ and } \sum_{q \text{ in } Q(n_j)} \frac{num(q, Q(n_j))}{num(Q(n_j))} r(q, n_i) = g/|V|,$$

whereas

$$\forall\ n_i,\ n_j \text{ in different groups, } Q(n_i) \text{ and } Q(n_j) \text{ have no queries in common and } \forall q \text{ in } Q(n_j),\ r(q, n_i) = 0.$$

CASE III: ASYMMETRIC SCENARIO. The shared content again belongs to t ($t > 1$) different categories. However, each node has content belonging to one category but poses queries for content belonging to a single different category. All categories have the same number of nodes (i.e., $|V|/t$ each) maintaining their content and querying their content, i.e., $|V|/t$ nodes are interested in (maintain) data of type i for all $1 \leq i \leq t$. Furthermore, all the nodes that maintain (query) content from a category maintain (query) the same portion of data of this category (i.e., $t/|V|$).

2.1 Stability

To determine whether a cluster configuration constitutes a Nash equilibrium, we need to ensure that the individual cost of each node is not smaller in any possible configuration that can result from the current one by changing only the strategy of this node (Ineq. (1) in [14]). To restrict the number of potential configurations, we rely on Lemmas 1 and 2 in [14].

CASE I: NO UNDERLYING CLUSTERING. We study the following cluster configurations.

CASE(I.A) In this case, there is just one cluster and all nodes belong to it. From Corollary 1 in [14], the only way a node n_i can change its strategy is by leaving this cluster and forming a cluster of its own, since remaining in the cluster and forming a cluster of its own does not improve its individual cost.

CASE(I.B) In this case, there are $|V|$ singleton clusters. The only way for a node n_i to change its strategy is to leave its own cluster and join k other clusters, where $1 \leq k \leq |V| - 1$.

CASE(I.C) The nodes form m *non-overlapping clusters* of the same size $|V|/m$. Consider a node $n_i \in c_j$. The available options for n_i for changing its strategy are to: (1) form a cluster of its own; (2) additionally to c_j, join k other clusters, where $1 \leq k < m$; or (3) leave c_j and join k other clusters.

CASE II: SYMMETRIC SCENARIO.

CASE(II.A) Same as Case (I.A).

CASE(II.B) The only option for n_i is to join k other nodes n_j, $1 \leq k \leq |V| - 1$ which belong either to the same group as n_i or to different ones. Since joining nodes from different groups does not reduce its recall or membership cost, a node only considers joining k nodes from its own group, thus, $k < |V|/g$.

CASE(II.C) In this case, we consider $m = g$ and that each of the m clusters contains nodes of a single group. Then, the individual node cost for each node $n_i \in V$ is equal to its membership cost, since the cost for computing queries outside its cluster is zero (there are no results for $Q(n_i)$ in nodes not in $V(s_i)$). The only option for n_i is to move to a cluster of its own, since joining any other clusters would not improve its recall or its membership cost.

CASE III: ASYMMETRIC SCENARIO.

CASE(III.A) Same as Case (I.A).

CASE(III.B) Similarly with Case (II.B), a node may join k other nodes, $1 \leq k \leq |V|/t$, that maintain content that belongs to the same category as its queries.

CASE(III.C) In this case, the nodes form $m = t(t-1)/2$ non-overlapping clusters of the same size $2|V|/t(t-1)$ such that half of the nodes in each cluster maintain content belonging to category t_i and pose queries for category t_j and the other half content of t_j and queries of t_i. The options for a node are to: (1) leave its cluster and form a cluster of its own, (2) leave its cluster and join k other clusters or (3) additionally to its own cluster, join k other clusters. Since from Lemma 2, the only clusters a node may consider are the ones maintaining content that it queries, $1 \leq k < (t-1)/2$.

2.2 Social Optimum

We study whether the stable configurations that we have previously considered achieve a social cost equal to the social optimum. We can acquire a rough bound of the social optimum by considering each node separately and evaluating its individual cost over all possible configurations. Then, by selecting for each node the minimum individual cost and adding these values, we obtain a bound for the minimum value of the social cost, i.e., for the social optimum. This estimation may be far from the actual value of the social optimum that can be achieved, since we are adding together individual costs that may correspond to different configurations and thus, the estimated social cost may refer to a configuration that cannot exist.

Lemma 3, in [14], allows us to reduce the number of possible configurations we need to consider for each node n_i to determine the minimum individual cost it may achieve. In particular, all configurations in which n_i belongs to more than one cluster do not have the minimum cost.

No Underlying Clustering. Since the game is symmetric, based on Observation 1, it suffices to minimize the individual cost of any node n_i. Furthermore, all configurations in which node n_i belongs to clusters with equal size yield the same individual cost for n_i as its recall loss and membership cost are equal regardless of the particular subset of nodes that belong to the cluster.

Based on the above, we compare our stable configurations with all possible configurations in which a node n_i belongs to one cluster of size $|V| - |V_0|$, where $0 < |V| - |V_0| \leq |V|$.

Case (I.A), i.e., all nodes form a single cluster, is stable for $\alpha \leq \frac{|V|-1}{\theta(|V|)-\theta(1)}$ (Table 1 in [14]). We compare the cost of Case (I.A) with all other possible configurations by considering a configuration such that n_i belongs to a cluster of size $1 \leq |V| - |V_0| < |V|$. We examine whether the individual cost for n_i for this configuration is larger or equal to the corresponding cost for Case (I.A), that is whether

$$\alpha\frac{\theta(|V|)}{|V|} \leq \frac{\alpha\theta(|V|-|V_0|)-|V_0|}{|V|}.$$

This holds when Case (I.A) is stable, i.e., for $\alpha \leq \frac{|V|-1}{\theta(|V|)-\theta(1)}$. Thus, we determine that when Case (I.A) is stable, the cost of the configuration is optimal. Analogously, Case (I.B) is stable and has a cost equal to the social optimum for $\alpha \geq \frac{k}{k\theta(2)-\theta(1)}$. Case (I.C) has a cost equal to the social optimum when the corresponding conditions in Table 1 in [14] hold for (I.C) to be stable. For this α, all three configurations have the same optimal cost.

Symmetric Scenario. In this case, any configuration that includes clusters with nodes from more than one group does not have an optimal cost, since its cost can be reduced if the cluster is split by separating the nodes from the different groups. Therefore, in this case we consider only configurations with $0 < |V| - |V_0| \leq |V|/g$.

For all $\alpha > 0$, Case (II.A) does not have cost equal to the social optimum, since it includes nodes from different groups in one cluster, while (II.B) has cost equal to the social optimum for $\alpha \geq \frac{kg}{k\theta(2)-\theta(1)}$, and (II.C) for $a \leq \frac{|V|-g}{\theta(|V|/g)-\theta(1)}$.

Asymmetric Scenario. Similarly to the symmetric scenario, we consider configurations with $0 < |V| - |V_0| \leq |V|/t + 1$.

Case (III.A) does not have a cost equal to the social optimum for any $\alpha > 0$, because as in (II.A), separating the different groups results in a configuration with lower social cost. Case (III.B) has cost equal to the social optimum for $\alpha \geq \frac{kt}{k\theta(2)-\theta(1)}$. Finally, Case (III.C) does not have a cost equal to the social optimum for any $\alpha > 0$, since, if, we remove from the cluster of node n_i the $\frac{|V|}{t(t-1)} - 1$ nodes that maintain content of the same category as the content of n_i, its individual cost is improved.

2.3 Load Balance

All three cases are by their definition (0)-size balanced, since all clusters contain the same number of nodes.

No Underlying Clustering. From Observation 1 in [14], Case I is also $(0,0)$-load balanced, since the content and the query workload is distributed uniformly among all nodes. Let us study whether a configuration with non equal sized clusters is stable. To this end, we consider Case (I.D), that extends Case (I.C), by assuming a configuration of m non-overlapping clusters each with a different size $|c_k|$, $1 < |c_k| < |V|$, $1 \leq k \leq m$. For this case to correspond to a Nash equilibrium, we have $\forall n_i \in c_k$, $\forall k' \neq k$:

$$\alpha(\theta(|c_k|) - \theta(|c_{k'}|)) \leq |c_k| - |c_{k'}| \qquad (1)$$

Since θ is increasing monotonously, it also holds that: $\frac{\theta(|c_k|)-\theta(|c_{k'}|)}{|c_k|-|c_{k'}|} = 1/\alpha$

In the general case, there may be stable configurations with non equal sized clusters depending on θ and the value of α. Thus, all stable configurations are not necessarily size or load balanced. According to Ineq. (1), we can evaluate the appropriate δ, δ_q and δ_r.

Symmetric and Asymmetric Scenario. Case (II.C) and (III.C) are (0)-size balanced but not $(0,0)$-load balanced, unless the total amount of query workload and content of each of the g and t categories respectively is the same. For Case (II.C), since each cluster maintains content from a single category and queries content only of the same category, we may consider each cluster separately. By partitioning each cluster to m non-overlapping clusters, we obtain results similar to Case (I.D). For Case (III.C), we derive the same results, if we split each of the $t(t-1)/2$ clusters in m sub-clusters, so that in each of the created sub-clusters half of the nodes maintain content belonging to category t_i and pose queries for category t_j and the other half content of t_j and queries of t_i.

Algorithm 1 Coordinated Protocol

$|V|$: number of nodes, $C = \{c_1, \ldots, c_n\}$: cluster set, $R = \{r_1, \ldots, r_n\}$: cluster representatives

```
 1: for all global events do
 2:    for all $r_i \in R$ do
 3:       send a game initialize request to all $n_j \in c_i$
 4:       for all $n_j \in V$ do
 5:          evaluate $gain(n_j)$
 6:          select randomly one of the representatives $r'_j$, $n_j$
             has received a request from
 7:          send to $r'_j$ an update request with $gain(n_j)$ for $S_{new}$
 8:       end for
 9:       send all update requests to all other $r_k \in R$
10:       sort update requests in non-increasing order of gain
11:    end for
12:    for all $gain(n_j)$ within $K$% of the list do
13:       $n_j$ updates its strategy from $s_{cur}$ to $s_{new}$
14:    end for
15: end for
```

3 The Coordinated Protocol In Detail

We now describe in detail the *coordinated protocol* where one node r_i at each cluster c_i serves as a cluster representative.

The protocol is presented in Alg. 1. Unlike the basic uncoordinated protocol, the coordinated one is triggered by any global event. In particular, the representatives are the ones that initiate the game after becoming aware of some event (line 1), by sending initialization requests to the nodes in their clusters (lines 2-3). Each node n_j re-evaluates its gain by considering all possible configurations S_j that differ only in s_j (line 5). Let S_{new} be the configuration with the best gain for n_j. Node n_j sends an update request to one representative r'_j along with the corresponding gain. The representative r'_j is selected randomly from the cluster representatives from which n_j has received an initialization request (lines 6-7). All update requests made by individual nodes are gathered by the representatives and exchanged among them so that each representative has all the requests (line 9). All requests are ordered by non-increasing value of gain (line 10) and the overall top-K percentage of them is granted (lines 11-13).

4 Additional Experiments

In this section, we present a number of additional experiments. First, let us present in detail our model. Each node is associated with a data category j and maintains documents belonging to it. The local query workload of each node is generated by first selecting a data category with probability $P(j)$ following a zipf distribution, and then a document d from this category with probability $P(d, j)$ following another zipf distribution within each category. The use of the zipf distribution in both cases assures that the derived query workload is such that popular data categories have more nodes interested in them and also popular documents also have more queries directed at them. We define $P_{n_i \in l}(d, j)$ as the probability of node n_i associated with category l posing a query about document d of category j as:

$$P_{n_i \in l}(d, j) = \begin{cases} (1-L)P(d,j), l \neq j \\ ((1-L) + L/P(j))P(d,j), l = j \end{cases}$$

The use of $P_{n_i \in l}(d, j)$ enables us to favor the data category l node n_i belongs to when generating its query workload, that is, increase the probability that a query is directed to a document of its own category by a factor of $(L/P(j)) * P(d, j)$. Parameter L which controls this factor is the measure of interest-based locality [25]. When $L = 0$ the category of the node is independent of the category of its query workload, i.e., there is no locality in its interests. Whereas, when $L = 1$ all queries belong to the same category as the node's content.

We consider three scenarios which correspond to the case studies we have studied theoretically. In the *symmetric scenario*, corresponding to CASE II, $L = 1$ and both queries and documents of each node belong to the same category. In the *asymmetric scenario* corresponding to CASE III, $L = 1$ but for a $j \neq l$ which is selected randomly from the remaining categories. That is, each node has documents from one category but poses queries for a single different category. The symmetric scenario exhibits maximum interest-based locality, while the asymmetric none. Finally, in the *random scenario* (no underlying clustering) corresponding to CASE I, $L = 0$.

TABLE 1: Workload cost for cluster formation

	WCost		
	FK	SSR	FSR
Symmetric Scenario			
i	10.04	10.14	10.12
ii	10.34	10.46	10.51
iii(a)	10.19	11.21	11.26
iii(b)	11.09	11.41	11.53
iii(c)	10.21	10.78	11.08
iv	10.09	10.09	10.09
v	10.09	10.09	10.09
Asymmetric Scenario			
i	914.05	978.45	976.12
ii	919.35	1015.91	927.32
iii(a)	918.2	1002.20	1050.10
iii(b)	922.25	1018.46	998.01
iii(c)	918.95	1014.12	1023.75
iv	924.25	1048.07	1043.25
v	918.75	1022.25	1018.16
Random Scenario			
i	11.33	11.33	11.33
ii	11.33	11.33	11.33

In this case, there is no interest-based locality and for each node, both its documents and queries are uniformly distributed from all categories.

4.1 Sensitivity Analysis

We first present experiments evaluating the behavior of the protocol with respect to the other tuning parameters besides the stopping condition ϵ, i.e., the threshold gain value lower than which a node does not issue any cluster update requests. To achieve a fair comparison between all protocols, we set for the experiments in this set $\epsilon = 10^{-2}$, since, as depicted in Fig. 2a and Fig. 2b in [14], the uncoordinated protocol with monitoring does not converge for smaller values.

For both protocols with FK, a smaller P_u reduces the number of moves, but increases the time it takes to reach stability since each node does not move every time it plays (Fig. 1a). In general, with partial knowledge, a larger P_u is required for ensuring that stable states are discovered within a reasonable amount of moves. Also, decreasing the update probability does not always guarantee that the number of moves is reduced. For example, both FSR and SSR require more moves for $P_u = 0.1$ than for 0.2. This is because if a node misses the opportunity to move to an appropriate cluster when it becomes aware of it, it may move to other suboptimal clusters many times before finding again the appropriate one. A similar behavior is achieved by tuning the *batch size*. Using quota in combination with the update probability reduces the overhead further (Fig. 1b).

An advantage of the tuning parameters is that they can be set on a per node basis. Observing the processing and communication overhead caused by its moves along with the associated fluctuations in its utility function, each node can tune the values of the parameters independently of the other nodes. For example, a high individual cost may cause the node to decrease its value of ϵ, while a large overhead to increase it. Similarly, if a node notices too many moves, it may decrease its update probability or quota, or if the node wants to react faster to changes, it may increase either or both of them.

The values reported in Fig. 2a and Fig. 2c in [14] comparing the uncoordinated with the coordinated protocol correspond to the value of the social cost, when

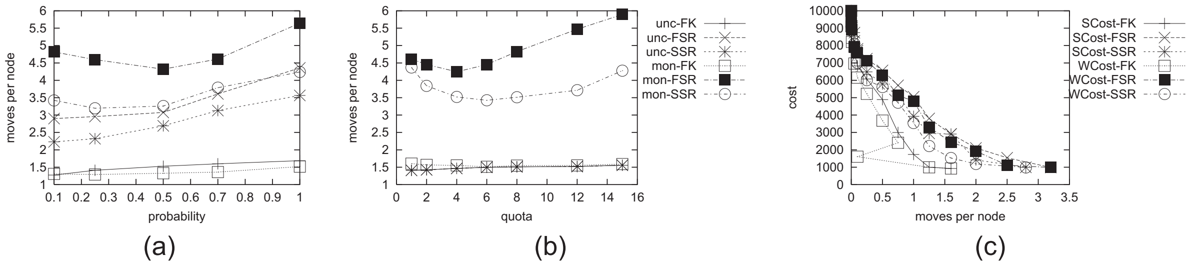
Fig. 1: (a) Varying update probability, (b) varying quota and (c) workload cost vs social cost.

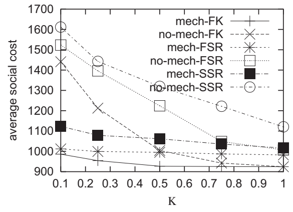
Fig. 2: Coordinated protocol: effect of mechanism.

the system stabilizes. We have also studied how the value of the social cost changes as the game progresses (Fig. 2). At the first steps of the game, the coordinated protocol reduces the social cost more drastically than the uncoordinated ones. The reason is that the coordinated protocol uses a mechanism by which the update requests of all nodes are ordered according by non increasing value of gain before granting the top-K percentage of them. This mechanism ($mech$) favors the most cost influencing nodes by granting their requests first, while the uncoordinated protocols treat all nodes as equals. Compared to not using this mechanism with the coordinated protocol (no-$mech$), and instead selecting randomly the K percentage of all requests that are granted, the use of the mechanism reduces the average social cost per step up to 10% (Fig. 2c in [14]).

4.2 Cluster Formation

We extend the evaluation of the cluster formation by including the workload cost in our measures (Table 1).

For the asymmetric scenario, since queries are not uniformly distributed among nodes, the social and the workload cost exhibit the larger differences. We consider how the two measures progress as the nodes make more moves. By adjusting the update probability P_u according to the node's demand levels, the workload cost ($WCost$) is reduced faster, while the social cost ($SCost$) that considers all nodes as equals decreases linearly (Fig. 1c) with the number of moves.

4.3 Updates

We first consider additional query workload update scenarios. In $WSc2$, k existing categories become popular, and in $WSc3$, k categories cease to be popular, i.e., are not queried anymore. In general, update scenarios that tend to form non uniform clusters, such as $WSc1$ where all update requests target the cluster(s) containing one specific data category, perform slightly worse than scenarios with requests evenly distributed among more clusters ($WSc2$).

We also consider updates of both the workload and the content. Such updates are common, since changes in the workload of a node, often entail changes in its content, i.e., if a node becomes interested in a new data category, then it will gradually acquire data of this category. At first, such an update increases the social cost as there are not enough data to satisfy the query workload of the updated nodes (Fig. 3c). Gradually, as they change more of their data, the updated nodes form a new cluster, thus, reducing the social cost.

We now consider topology updates, i.e., nodes joining and leaving the system. When new nodes join the system, they initially increase the social cost. New nodes that enter the system pose queries by contacting a few random nodes of the system they have become aware of while entering. As they acquire results for their queries, the new nodes are gradually informed about the clusters in the system. Thus, by re-evaluating their individual cost based on the new knowledge they acquire with time, they select an appropriate cluster to join (Fig. 3d). The lack of knowledge does not significantly affect this scenario as all nodes operate with partial knowledge in the beginning regardless of the protocol variation used; FK is only marginally faster.

When nodes leave the system, the social cost is reduced as the number of nodes is reduced. The individual node cost is not affected significantly in this case, i.e., the average individual node cost remains around the same, as results that belonged to nodes that have left the system are not considered as recall loss. No immediate adaptation is required but if a large percentage of the members of a specific cluster leave then the individual cost of its members may be more drastically affected. The remaining members may move to improve recall, or nodes from other clusters may move to this cluster due to its low membership cost.

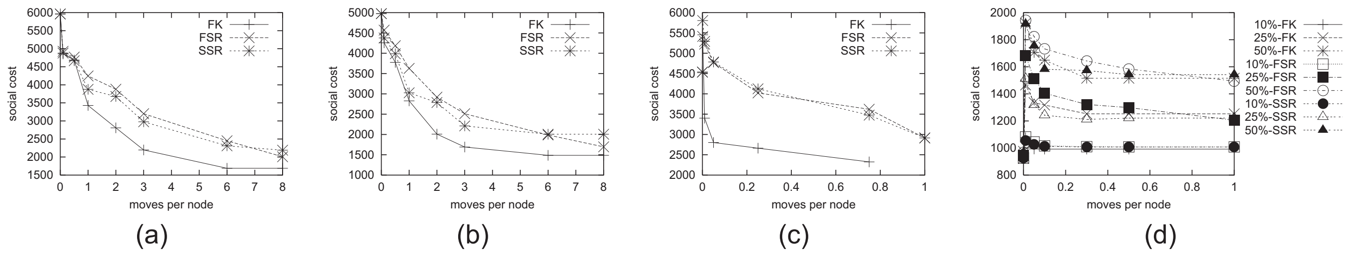

Fig. 3: Updates: (a) $WSc2$ and (b) $WSc3$ with moves, (c) workload and content and (d) nodes joining.

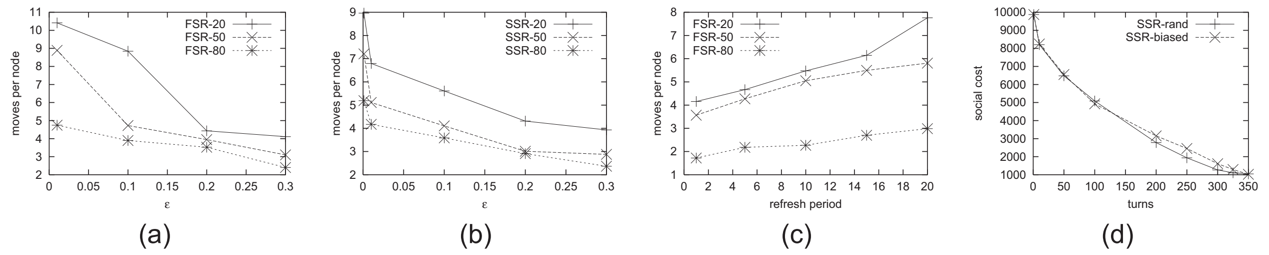

Fig. 4: (a) FSR, (b) SSR, (c) varying refresh period and (d) biased routing.

4.4 Degree of Local Knowledge

In the last set of experiments, we study the influence of the lack of full knowledge further by considering different degrees of knowledge for FSR and SSR. We use the setting of the first set of experiments, i.e., an asymmetric scenario with an initial configuration where each node forms a cluster of its own.

We first consider FSR protocols with $C(n_i)$ corresponding to 20%, 50% and 80% of C_{cur}, while we fix the refresh period at 10 queries and measure the overhead in terms of moves for varying ϵ (Fig. 4a). The larger percentage $C(n_i)$ covers, the less moves are required to reach stability. Thus, lower ϵ values are appropriate that also reduce the social cost for the same number of moves.

Similarly for SSR, we consider protocols such that $C(n_i)$ and $RC(q, n_i)$ for each query q and each node n_i vary from 100%-20%, 100%-50% and 100%-80%. Figure 4b illustrates the social cost for each protocol. $C(n_i)$ is fixed to 100% of C_{cur} to attain a fair comparison with FSR so that the corresponding protocols entail the same routing overheads. SSR reaches a stable state with less moves than FSR, but the social cost is slightly higher for the same ϵ. By selecting for SSR lower values for ϵ, we may achieve a social cost similar to the one in FSR, but with an increased number of moves. If we also decrease the percentage of clusters $C(n_i)$ covers, the performance of SSR becomes worse compared to FSR, but the routing cost for each query is reduced.

We also study how the refresh period affects FSR by varying it from 1 to 20 queries. For small periods (< 5), the protocols behave similarly to a corresponding SSR, while for a period equal to 1, FSR becomes the corresponding SSR (Fig. 4c). With respect to the social cost, as the period increases, we have a slight improvement as the protocols evaluate more representative values for each cluster they are aware of.

Finally, we study using bias in SSR when selecting $R(n_i, q)$. We utilize a zipf distribution based on previous query results to select $R(n_i, q)$ (*SSR-biased*) and compare it to a protocol with randomly selected $R(n_i, q)$ (*SSR-rand*). We consider a 100%-50% SSR variation and measure the social cost for progressing turns (Fig. 4d). The social cost is improved compared to a random selection but the turns required are almost 10 times more.

4.5 Confidence Intervals

In all experiments, we report the average value of 100 runs with different initial configurations so as to cope with statistical errors. To gain a better understanding of the variance our results exhibit, we also measured confidence intervals for the mean values of our runs with 95% confidence.

In general, we do not observe large intervals, while the most variation is exhibited with partial knowledge where the initial configuration influences our results more. The worst case for the measured social cost appears in the cluster formation experiment when using random clustered topologies as our initial configuration (case iii) and partial knowledge (Table 2 in [14] and Table 1). In this case, the social cost varies between a [1031.205, 1031.265] with 95% confidence. The number of clusters in the same experiment is the measure that exhibits the greatest variance within an [41, 46] interval. The other measures behave similar to the social cost, i.e., the worst case for the number of moves per node is an [2.693, 3.112] interval and similarly for the workload cost the length of the interval does not exceed 0.5. For δ_s, we observe even smaller fluctuations, with intervals of length in the order of 10^{-2}.

5 ADDITIONAL RELATED WORK

There has been a lot of research regarding network creation games. In [19], the authors show that allowing nodes to act completely freely performs much worse than collaboration, and prove that even a static p2p system of selfish nodes may never reach convergence. This result agrees with our findings that show that only in specific scenarios, we reach stability. [6] considers the optimal topology in terms of the social cost for a variation of the game where the cost includes: (a) the cost to locate an item expressed in number of hops, (b) a constant cost of servicing a request, (c) the cost to route a request as the number of paths to the servicing node and (d) the cost to maintain the overlay in terms of the number of neighbors of the node.

On another note, in this paper, we have studied partial knowledge experimentally. There has been some very recent work on modeling partial knowledge using Bayesian games [1]. In a Bayesian game, agents have only a local view of the system and cannot coordinate their actions on the global state of the system. The lack of the global view is referred to as Bayesian ignorance. An interesting direction would be to define our cluster game as a Bayesian one and use Bayesian ignorance to quantify the effect of partial knowledge theoretically. An importance difference is that in our case, the local view of an agent (i.e., node) includes some partial knowledge of the strategies of the other agents in its cluster.

There has been also a large body of research on the formation of clustered overlays in p2p systems towards efficient query processing. In most cases, the focus is on cluster formation, while the adaptation of the overlay is not addressed. None of the proposed clustering methods takes a game-theoretic approach.

In [2], nodes are partitioned into topic segments based on their content. A fixed set of M clusters with centroids that are globally known is assumed, each one corresponding to a topic segment. Clusters are formed in [27] based on the semantic categories of the data of the nodes; the semantic categories are predefined. Similarly, [8] assumes predefined classification hierarchies based on which queries and data are categorized. Instead of predefined categories, [11] uses a learning approach that by generalizing the shared data, learns the semantic categories they belong to and then uses those for clustering. In [4] the clustering process takes into account besides predefined semantic categories also the proximity of the nodes in the physical network to improve routing efficiency inside a cluster, but relies on the use of super-peers that act as cluster leaders.

In [15], clustering is based on data schemas and predefined policies provided by human experts. Clustering based on the similarity of schemas globally known is also used in [21], in which a two step approach incrementally forms semantic clusters by first appointing new peers to the most similar clusters and then enabling them to choose the most similar neighbors. In [9], clustering is first applied on the documents of each node, and then recursively on the derived feature vectors by selected node representatives. While this approach does not assume predefined categories, it still requires the use of cluster representatives unlike our uncoordinated protocol.

Besides clustering based on the content of nodes, clustering based on other common features, such as the interests of nodes [13], is possible. In [7], nodes maintain sets of guide rules, which are formed by the users either explicitly based on their interests, or implicitly through query history, thus defining semantic clusters. A somewhat different approach to clustering is taken in pSearch [26] that maps node documents on a DHT, based on their term vectors and exploiting only the most important terms. Thus, semantically related documents are "clustered" in the DHT, limiting the search space.

In [12], a supernode-based architecture is proposed in which nodes with common interests are organized based on their caches. The paper exploits the idea of [25], and since it is based on caches it implicitly addresses the issue of cluster adaptation, but does not focus on it. Cache sharing in conjunction with methods for summarizing the cache content for efficient searching have been shown to improve performance in many distributed settings [10]. A work that focuses on adaptation issues is presented in [22], in which different rewiring techniques for the adaptation of semantic links are investigated. The work is complementary to our own as it focuses on the routing protocols utilized to discover new peers, which is an issue orthogonal to our game protocol.

Yet another example of clustering in p2p systems is that induced by the tit-for-tat of choking algorithm used as the peer selection algorithm in the BitTorrent protocol [16]. The BitTorrent protocol has established swarming, i.e., parallel download of a file from multiple peers with concurrent upload to other requesting peers. Experimental results have demonstrated that the choking algorithm leads to the formation of clusters of peers that have similar upload capacities among the torrents, i.e., those peers cooperating to load the same content. The properties of BitTorrent are used in [24] to support n-way broadcasting where each one of n overlay nodes must push a file to all other n-1 peers, as well as pull the n-1 files pushed by these other peers. An optimized overlay is constructed upon which swarming is performed. Each node selects other nodes to be in its peer-set and establishes connections to them so that to maximize either the available bandwidth to the slowest destination (max-min) or the aggregate output rate (max-sum).

An interesting approach where peers in unstructured p2p use erasure coding is presented in [18]. Files are split into equally sized pieces (chunks). Peers are responsible for establishing bi-directorial links to the peers storing chunks of their data. A peer that needs to store chunks is required to offer storage space of similar size. There is a global rank of peers based on their quality (on line availability and bandwidth capacity) and amount of resources they provide, called profile. A non-cooperative game introduced where peers selfishly choose to establish links with remote peers with good quality and also minimize the cost they bear for storing copies. Again, it is proved that the system reaches an equilibrium when it is stratified, i.e., clustered in the following sense: peers with similar profiles cooperate by building bilateral links with each other. Similarly, data availability is studied in [23], in a distributed data storage system where instead of erasure coding, whole files are replicated among the peers. The problem is studied for both a centralized and a decentralized scenario by modeling selfish peers that aim to maximize their data availability, while minimizing the amount of others data they store. The results for the decentralized case show the formation of implicit clusters as peers replicate only with similarly available peers.

Besides clustering, other approaches to improve search in unstructured p2p systems include replication as well as more efficient searching than flooding such as random walks; which are issues not addressed in this paper. In this context, various approaches have been proposed (e.g., [5], [17]).

Finally, another interesting form of overlay networks are those formed to connect publishers and subscribers in publish/subscribe systems. In publish/subscribe systems, subscribers express their interests in specific pieces of information often described as topics and get notified when publishers produce items that match their interests. Clustered overlays have been proposed to improve the performance of such systems. For example, [3] proposes a distributed clustering algorithm that utilizes correlations between user subscriptions to dynamically group topics together into virtual topics or topics clusters. It would be interesting to apply a game theoretic approach to clustered overlays in this setting as well.

Besides modeling clustered overlays, a game-theoretic approach could be applied to consider other types of publish/subscribe overlay networks as well. For instance, [20] considers the following problem in this area: given a collection of nodes and their topic subscriptions, connect the nodes into a network with low average and maximum degree such that for each topic, the network induced by the nodes (i.e., subscribers) interested in this topic is connected. A polynomial time parameterized sublinear approximation algorithm is proposed for this problem.

APPENDIX

Lemma 1 in [14]: In any stable cluster configuration, there are no clusters c_i, c_j such that $c_i \subseteq c_j$, $i \neq j$.
Proof. Let S be a cluster configuration and c_i, c_j two clusters in C such that $c_i \subseteq c_j$. Consider a node n_k, $n_k \in c_i$. Clearly, $n_k \in c_j$. Let the individual cost of n_k be: $icost(n_k, S) = \alpha\gamma + \delta$, where γ is the membership cost for n_k when following strategy $s_k \in S$ and δ the respective recall it loses from the nodes that do not belong to $V(s_k)$. Assume for the purposes of contradiction that S is a stable configuration, then n_k can not select a strategy that would reduce its cost. Let us examine the strategy $s'_k = s_k - \{c_i\}$. Let S' be the configuration resulting by replacing s_k with s'_k in S. Then, $icost(n_k, S') = \alpha(\gamma - \frac{\theta(|c_i|)}{|V|} + \delta) < icost(n_k, S)$. The recall part of the cost function remains the same, because $V(s_k) = V(s'_k)$. Thus, n_k can reduce its cost by selecting the strategy s'_k, and therefore S is not stable, which contradicts our assumption.□

Lemma 2 in [14]: For any strictly increasing function θ, there is no stable configuration in which $\exists c_i$, $|c_i| > 1$, $c_i \in s_j$ and $\sum_{q \text{ in } Q(n_j)} \sum_{n_k \in c_i} r(q, n_k) = 0$.
Proof. Let S be a cluster configuration, n_j a node and c_i a cluster of size $|c_i| > 1$, such that $c_i \in s_j$ and $\sum_{q \text{ in } Q(n_j)} \sum_{n_k \in c_i} r(q, n_k) = 0$. If s_j=$\{c_i\}$, n_i can improve $icost(n_i, S)$ by moving to a cluster of its own, since $\theta(|c_i|) > \theta(1)$ and the recall loss remains the same. If $c_i \subset s_j$, n_i can improve $icost(n_i, S)$ by selecting strategy $s'_j = s_j - \{c_i\}$, which reduces its membership cost without affecting its recall loss. Thus, S is not stable.□

Proposition 1 in [14]: A pure Nash equilibrium does not always exist for the cluster formation game.
Proof. Let us consider a simple scenario of two nodes n_1 and n_2. Let us denote as r_{21} the results of $Q(n_1)$ maintained by n_2, and r_{12} the results of $Q(n_2)$ maintained by n_1, $0 \leq r_{21}, r_{12} \leq 1$. Let $C = \{c_1, c_2\}$ be the clusters in the system. Using Lemma 1, the following cluster configurations are possible: $n_1 \in c_1$ and $n_2 \in c_2$, described by $S_1 = \{\{c_1\}, \{c_2\}\}$, $n_1 \in c_2$ and $n_2 \in c_1$, described by $S_2 = \{\{c_2\}, \{c_1\}\}$ and both $n_1, n_2 \in c_1$ or c_2 described by $S_3 = \{\{c_1\}, \{c_1\}\}$ and $S_4 = \{\{c_2\}, \{c_2\}\}$, respectively. Table 2 summarizes the payoff (cost) table for this two-player game. Since configurations S_1 and S_2 are symmetric, let us examine the first one. If n_1 moves to cluster c_2, then the system configuration becomes $\{\{c_2\}, \{c_2\}\}$, that is, configuration S_4, and the cost for both n_1 and n_2 becomes $\frac{\alpha\theta(2)}{2}$. Therefore, if $\frac{\alpha\theta(2)}{2} < \frac{\alpha\theta(1)+2r_{21}}{2}$, or $\frac{\alpha\theta(2)}{2} < \frac{\alpha\theta(1)+2r_{12}}{2}$, then configuration S_1 is not a Nash equilibrium. Let us consider now the configuration S_3 (S_4 is symmetric) where both nodes belong to the same cluster. If n_1 moves to an empty cluster then its cost becomes $\frac{\alpha\theta(1)+2r_{21}}{2}$. Thus, as with S_1, S_3 is not a Nash equilibrium if $\frac{\alpha\theta(1)+2r_{21}}{2} < \frac{\alpha\theta(2)}{2}$ or $\frac{\alpha\theta(1)+2r_{12}}{2} < \frac{\alpha\theta(2)}{2}$. Based on these observations we can deduce that for $\min(r_{12}, r_{21}) < \frac{a}{2}(\theta(2) - \theta(1)) \leq \max(r_{12}, r_{21})$ none of the four possible configurations is stable, and thus, a Nash equilibrium does not exist.
For instance, if we consider the worst case for the θ function is the linear function, in which case, we have that for α such that $\min(r_{12}, r_{21}) < \frac{a}{2} < \max(r_{12}, r_{21})$, the two player game has no Nash equilibrium. □

Lemma 3 in [14]: Let θ be a function such that for any x, $x_j \in N$, $1 \leq j \leq k$, if $x < \sum_{j=1}^{k} x_j \Rightarrow \theta(x) < \sum_{j=1}^{k} \theta(x_j)$,

TABLE 2: Payoff table

	n_2 joins c_1	n_2 joins c_2
n_1 joins c_1	$\frac{\alpha\theta(2)}{2}, \frac{\alpha\theta(2)}{2}$	$\frac{\alpha\theta(1)+2r_{21}}{2}, \frac{\alpha\theta(1)+2r_{12}}{2}$
n_1 joins c_2	$\frac{\alpha\theta(1)+2r_{21}}{2}, \frac{\alpha\theta(1)+2r_{12}}{2}$	$\frac{\alpha\theta(2)}{2}, \frac{\alpha\theta(2)}{2}$

then for any node n_i, a configuration S with minimum $icost(n_i, S)$ is such that n_i belongs to just one cluster.
Proof. Let S be a configuration where a node n_i belongs to $l > 1$ clusters of sizes $|c_1|, |c_2|, \dots, |c_l|$ and θ a function such that if $x < \sum_{j=1}^{k} x_j \Rightarrow \theta(x) < \sum_{j=1}^{k} \theta(x_j)$. In this case, the membership cost of n_i for S is always larger than the membership cost for a configuration S' where n_i belongs to a single cluster that includes the union of the nodes of clusters c_1 to c_l. Since the recall cost of n_i for S' is the same as for S, $icost(n_i, S') < icost(n_i, S)$. $\square$

REFERENCES

[1] N. Alon, Y. Emek, M. Feldman, M. Tennenholtz. Bayesian ignorance. In *PODC*, 2010.

[2] M. Bawa, G. Manku, and P. Raghavan. SETS: Search enhanced by topic segmentation. In *SIGIR*, 2003.

[3] R. Boim and T. Milo. Enriching topic-based publish-subscribe systems with related content. In *SIGMOD*, 2008.

[4] J. Bo. Heterogeneity-Aware Group-Based Semantic Overlay Network for P2P Systems. In *WISM*, 2009.

[5] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham and S. Shenker. Making gnutella-like P2P systems scalable In *SIGCOMM*, 2003.

[6] N. Christin and J. Chuang. On the cost of participating in a peer-to-peer network. In *IPTPS*, 2004.

[7] E. Cohen, A. Fiat, and H. Kaplan. Associative search in peer to peer networks: Harnessing latent semantics. In *INFOCOM*, 2003.

[8] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems, Technical Report, Computer Science Department, Stanford University, 2002.

[9] C. Doulkeridis, K. Norvag, and M. Vazirgiannis. Desent: decentralized and distributed semantic overlay generation in p2p networks. *JSAC*, 25(1):25–34, 2007.

[10] L. Fan, P. Cao, J. M. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. In *IEEE/ACM Trans. Netw.* 8(3): 281-293, 2000.

[11] A. Fast, D. Jensen, and B. N. Levine. Creating social networks to improve peer-to-peer networking. In *KDD*, 2005.

[12] P. Garbacki, D. H. J. Epema, and M. van Steen. Optimizing peer relationships in a super-peer network. In *ICDCS*, 2007.

[13] M. Khambatti, K. Ryu, and P. Dasgupta. Efficient discovery of implicitly formed peer-to-peer communities. *IJPDSN*, 5(4):155–164, 2002.

[14] G. Koloniari and E. Pitoura. A game theoretic approach to the formation of clustered overlay networks. Main paper.

[15] A. Loser, F. Naumann, W. Siberski, W. Nejdl, and U. Thaden. Semantic overlay clusters within super-peer networks. In *DBISP2P*, 2003.

[16] A. Legout, N. Liogkas, E. Kohler and L. Zhang. Clustering and sharing incentives in BitTorrent systems. In *SIGMETRICS*, 2007.

[17] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS*, 2002.

[18] P. Michiardi and L. Toka. Selfish neighbor selection in peer-to-peer backup and storage applications. In *Euro-Par*, 2009. endthebibliography

[19] T. Moscibroda, S. Schmid, and R. Wattenhofer. On the topologies formed by selfish peers. In *PODC*, 2006.

[20] M. Onus and A. W. Richa. Parameterized maximum and average degree approximation in topic-based publish-subscribe overlay network design. In *ICDCS*, 2010.

[21] W. Penzo, S. Lodi, F. Mandreoli, R. Martoglia, and S. Sassatelli. Semantic peer, here are the neighbors you want! In *EDBT*, 2008.

[22] P. Raftopoulou, E.G.M. Petrakis, and C. Tryfonopoulos. Rewiring strategies for semantic overlay networks. *JDPD* 26: 181-205, 2009.

[23] K. Rzadca, A. Datta, and S. Buchegger. Replica placement in p2p storage: complexity and game theoretic analyses. In *ICDCS*, 2010.

[24] G. Smaragdakis, A. Bestavros, N. Laoutaris, J. W. Byers, P. Michiardi and M. Roussopoulos. Swarming on optimized graphs for n-way broadcast. In *INFOCOM*, 2008.

[25] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM*, 2003.

[26] C. Tang and Z. Xu and M. Mahalingam. pSearch: information retrieval in structured overlays. *Computer Communication Review*, 33(1):89–94, 2003.

[27] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards high performance peer-to-peer content and resource sharing systems. In *CIDR*, 2003.