Optimal decentralized formation of k-member partnerships (May 9, 2010)

Anna Chmielowiec, Maarten van Steen
Dept. of Computer Science
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
$\{ania|steen\}$@*cs.vu.nl*

***Abstract*—To keep pace with constantly changing markets, many companies are seeking strategic partnerships. In this paper, we assume that a company can electronically provide a profile of the product or service it has to offer. This profile is described in such a way that potential partners can assess the fitness of the company for eventually teaming up. We concentrate on the fully decentralized optimal formation of teams consisting of k members. This problem boils down to developing a decentralized, efficient algorithm for solving a variant of the maximal weighted k-subgraph problem. We provide a first solution, along with an assessment of its performance, thereby concentrating on the feasibility of an actual embedding in real-world scenarios consisting of thousands of companies. In particular, any solution should be highly adaptive when new or fresh information concerning potential partners comes available.**

***Keywords*-strategic alliances; product bundling; k-clique matching; variable neighbourhood search; self-stabilization;**

I. Introduction

For the rapid development of innovative elaborate products, companies are increasingly engaging themselves in different forms of strategic partnerships. Apple and Nike are a good example. In 2006 the two companies announced the *Nike+iPod Sport kit* — a two-piece sensor-based device that, when connected to an iPod, allows runners to track their workout data such as elapsed time, pace and distance covered as they are jogging and listening to music. The creation of this product was possible thanks to Nike's experience in the design of sportswear and understanding of a runner's needs and behaviour combined with Apple's expertise in digital and software technology.

These two companies are not the only ones that have seen benefits in mutual collaboration. Other firms, not only big corporations but also small and medium enterprises, are spotting the potential of this kind of cooperation for harnessing the competencies and resources of each participating party in pursue of a common objective. However, the very first challenge they have to face is finding suitable business partners.

The choice of business partners can have a significant impact on the success of a cooperation. As a consequence, the ability to discover and accurately assess the prospective collaborators is of foremost importance. Yet, the task of selecting partners is not easy and it becomes even more complicated if the company is looking for more than one partner for a specific venture. In such a case it has to evaluate not only the fit between itself and a prospective partner but also between the envisaged partners themselves.

Ironically, the advances in ICT and Internet have made the discovery of best possible partners more difficult. The current communication technology makes it potentially possible to team up with companies from all over the world. However, at the same time this means that the choice is much greater, which in many cases makes the decision process much more complex.

Furthermore, we need to realize that businesses are forced to respond ever faster to a constantly changing market, which is especially apparent in the sector of e-services. This imposes a special character on the goals of collaborations. Creating new products from scratch might be too risky and too time-consuming. Instead companies may integrate already existing products into one to create a joint offering. This "sale of two or more separate products in one package" is called bundling [1] and has been heavily researched in the marketing, economics, and law literature for over 50 years. Along with an increase in the number and variation of electronic goods (such as electronic newspapers, e-books, audio, and video) as well as electronic services that can be ordered and provisioned online (think, e.g., of streaming-media services), bundling has gained more interest also in the information and computer science community.

Partnerships should not only deliver the results quickly in order to gain the competitive advantage for their participants. Also the process of partnership creation should not be too time-consuming, as the competitors may seize the opportunity sooner. Yet, searching for "perfect" business partners might pose a real challenge for small companies as they lack the knowledge about which firms are there that have suitable capabilities and resources. Outsourcing the search to a specialized consulting company may not be a viable option due to high costs, nor is making use of only an in-house expert. An attractive alternative could be an Internet-based system capable of finding prospective partners in an automated way. As past years have shown, companies little by little are expanding the use of Internet and ICT into the growing number of their internal and external business activities. Therefore, such a system could find willing adopters, especially among e-services providers.

We believe that such a system should be decentralized — organized in a network of nodes, each representing a profile of one company. The main reason for avoiding a centralized implementation lies in the ownership that would be concentrated in one hand. As a result, the decision-making power would reside with the owners, raising concerns of trust. Moreover, the centralized solution would most likely be a paid service akin to the usually expensive consultancy firms.

This leads us to the following outline of the requirements for a *partnership formation service*:

- **Clarity** The rules by which the potential partners are selected should be clear and not too complicated.
- **Fairness** The service, while matching prospective partners, should respect the interests of all the companies that are represented by nodes in the system.
- **Minimal requirements** Use of the service should not pose high computational or memory requirements on the nodes.
- **Ease of deployment** Companies should be able to add their nodes to the system at any time without using complicated bootstrapping mechanisms.
- **Robustness** The impact of nodes entering or leaving the system should be minimal. Moreover, the service should be able to recover gracefully from the situations of many nodes leaving the service at the same time.
- **Scalability** The performance of the service should not be heavily effected by a growing number of nodes.

We will address most of this requirements and to the others we will provide a guidance for a plausible implementation.

Contributions

In this paper we present a self-stabilizing protocol for finding a matching between potential business partners. Our protocol is a generalization of the distributed algorithm for solving the weighted matching problem proposed by Manne and Mjelde [2]. However, our extension of this algorithm may impose a prohibitively high computational load on participating nodes. To alleviate this problem, we also present a heuristic that significantly reduces this load. We evaluate the performance of our protocol both with and without our proposed heuristic. Lastly, we discuss an example of a possible application for our protocol in the form of a network consisting of companies each of which is trying to select the best group of prospective partners for some joint project. We show how our protocol can be used in this kind of a setting to assist the formation of multilateral partnerships in a completely decentralized fashion.

Paper Overview: The remaining part of the paper is structured as follows. In the next section, related work is discussed. In Section III, we describe the system model for discovery of potential sets of partners. Section IV presents in detail the self-stabilizing distributed algorithm and heuristics used to limit the computational complexity. Section V provides the experimental evaluation of the protocol based on simulations followed by the last two sections containing discussion, plans for future work and conclusions.

II. Related Work

Business partnerships such as strategic alliances and co-branding collaborations have been a heavily studied topic in economics and marketing for years. The most pressing question that researchers have been trying to answer is what makes this kind of ventures successful. Many papers [3]–[5] stress the importance of the choice of appropriate partners, often presenting the guidelines for selecting the right business partners. For example, Prince and Davies [6] list four criteria that a company has to take into consideration while choosing a co-branding partner. Their very first recommendation is to analyze the compatibility between the firms in terms of both product fit and brand fit. Also Bronder and Pritzl in [4] list *fundamental fit*, *strategic fit*, and *cultural fit* as important factors in selection of business partners for strategic alliances.

From our perspective an important question is whether it is realistic to assume that the assessment of business partner fit can be automated. An important concept in this respect is that of *brand image*, defined as “perceptions about a brand as reflected by the brand associations held in consumer memory” [7]. Some preliminary research on automated partner fits has recently been done. In [8], Vermeulen shows how the World Wide Web can be used to assess brand images and how obtained measurements can be further utilized to discover promising brand alliances. He extracts the number of co-occurrences of a brand name with other brand names to compute the strengths of associations between them and to create a *competitive image* of a brand. Similarly, the strengths of associations between a brand name and some preselected attributes are computed to form *personality, evaluative, and semantic images*. All of these four images are later used by Vermeulen to predict the chances for successful co-branding ventures between pairs of companies. This is done by simply assessing the level of structural equivalence between the corresponding images of the two companies.

The ideas of *business ecosystems* [9], *network economy* [10] or *business webs* [11] have drawn attention to the structural organization of enterprises and markets, and the changes such organizations are undergoing. There is an irrevocable shift of focus from a single company or traditional value chains to the networks of companies interacting with each other. Van Heck et al. [12] carve a future scenario in which such networks become more agile. They foresee that companies operate in what they call a *pick, plug, and play* fashion, dynamically creating linkages with each other to provide complex, bundled goods and services. Digital technology plays a pivotal role in facilitating this constant re-organization of business connections.

This has laid foundations for research expanding into computer science. For example, Blau et al. [13] present a formal framework for modeling what they call *service value networks*. Such networks bring together legally independent providers of diverse services and facilitate flexible composition of complex services triggered directly by customers requests.

Opposed to service value networks research that focuses on creating a platform for service composition, our protocol stays more in line with the smart business networks concept of swarm-like organization of enterprises and contributes to making a step in the direction of the automated formation of linkages between prospective business partners.

Our matching protocol can be viewed as a protocol for the completely decentralized construction of an overlay network. The resulting topology partitions the network into fully connected groups of exactly k nodes. The formation of these groups emerges from the decisions of nodes based only on their local view of the whole network to evaluate the fitness of the group. Other protocols for fully decentralized overlay construction also exist. Approaches closely related to our work can be found, for example, in [14] or [15]. In [14], [16] Jelasity et al. present a protocol (called T-MAN) that constructs different types of overlay topologies depending on a ranking function used by nodes to compose neighbour lists. Similar to this protocol is VICINITY proposed by Voulgaris and van Steen in [15] which they apply, as an example, to the construction of an overlay that reflects the semantic proximity of nodes. Both T-MAN and VICINITY are gossip-based protocols, organized in a way that is akin to our matching protocol.

III. System Model

Throughout this paper, we concentrate on companies that can represent their business by means of a Web service, which, in turn, is available at a server dedicated to that company. Each company is said to be represented by a dedicated *node* in the Internet. Note that our approach does not necessarily imply that a company actually *offers* such a service as their product, but rather that each company can be *automatically* contacted and queried for what it has to offer. To this end, we assume that a description of a company is readily available, and in such a way that its relevance or usefulness can be evaluated by a contacting party. Such an evaluation is crucial in order to establish to what extent two or more parties are fit for a collaboration.

In our model, we express such a fitness in terms of a nonnegative *weight* between two parties A and B. The higher two parties assess that they fit, the higher the weight. For now, we adopt a simple approach in which the mutual fitness is symmetric: a collaboration between two parties is valued the same by both. We return to this symmetry later in our discussion. Moreover, the fitness of a collaboration between multiple parties is a combination of the fitness estimations between all the pairs of the companies.

To discover other parties, we assume the existence of a *partner discovery service*. There are many ways in which such a service can be realized. A traditional one consists of a broker that has become known to various parties through an external channel. In our view, a much better approach is to consider a fully decentralized partner discovery service realized by means of, for example, an epidemic protocol. The feasibility of such a decentralized service as been demonstrated by Jelasity et al. [17].

An epidemic-based partner discovery service is implemented by letting each node maintain what is called a *partial view* of the entire system. A partial view is a (relatively small) list with references to other nodes. Periodically, each node randomly selects a node from its view and initiates an exchange, yielding that references are passed back and forth between the two nodes. The effect is that partial views will appear to continuously consist of a uniform, random sample of all nodes in the system.

To select the best parties for collaboration, we let every node maintain a much larger list of other potential partners, a list which is continuously updated with new (and better fitting) entries obtained from the partner discovery service, while entries that have been evaluated to have a low fitness may be evicted. Again, to keep matters simple, we assume that entries are never evicted unless a node discovers that a referenced peer is no longer part of the system.

With this approach, we are capable of running fully decentralized algorithms, which logically operate at a layer on top of the partner discovery service. In particular, in the next section we describe a decentralized algorithm that would allow nodes to arrange themselves into prospective partnerships.

IV. A Matching Protocol

The problem of companies finding prospective partnerships can be translated into a well-known problem from graph theory. If we look at the nodes as a vertices in the graph, then the weight of an edge between a pair of any two vertices represents the fitness of these two nodes for a potential collaboration. Moreover, each clique of k vertices represents a potential k-party partnership and the weight of this clique depicts the overall evaluation of the cooperation aptitude of the firms that are part of this clique. We assume that all companies in the system are interested in participating in exactly one[1] partnership at a time, for

[1]If a company is considering creating more than one partnership, it could represent itself with more than just one node. However, in our current work we neglect neglect issues related to such multiplication of nodes; such as, company's participation in two different partnerships that may raise conflicts of interest or company's attempt to exploit the service and its other participants by putting an excessive number of nodes into the network.

Table I
NOTATION USED

Symbol	Meaning
$N(v)$	set of all neighbours of v
$\mathbf{C}[v]$	set of $k-1$ neighbours with which v wants to create a clique
$w_v(C)$	weight of the clique consisting of v and nodes in C
$\mathbf{w}[v]$	the weight $w_v(\mathbf{C}[v])$
$attr_v(C)$	the attractiveness of clique C for node v (see text)

example, due to limited resources and we focus solely on the formation of these potential strategic structures. We thus abstract from the details of particular projects that firms might intend to realize. We express the problem of discovering partners in terms of what is known in graph theory as a *(edge) weighted k-clique packing* problem.

Definition 1 (Weighted Clique Packing): Given a graph $G = (V, E)$ a *clique packing* is a disconnected subgraph of G whose components are cliques. If edges in G have assigned nonnegative weights, we define the weight of an *(edge) weighted clique packing* as the sum of the weights of all cliques from this packing. The *(edge) weighted clique packing problem* concerns finding the clique packing of maximum weight in G.

The fact that the cliques in a clique packing are disconnected reflects our initial assumption that the companies limit themselves to the participation in at most one partnership. Obviously, each company would attempt to form the most promising collaboration. Thus, each node would strive to maximize the weight of the clique it is part of in a clique packing. As a consequence, we might be less interested in the global maximum and more in a local greedy algorithm that gives a suboptimal global solution but maximizes the individual prospects of nodes.

If we restrict ourselves to a situation in which all companies are looking for the same number of partners, we can reduce our problem to the *(edge) weighted k-clique matching* problem.

Definition 2 (Weighted k-Clique Matching): Given a graph $G = (V, E)$ with nonnegative edge weights, a *k-clique matching* is a clique packing where each clique has exactly k vertices. The weight of the *k-clique matching* is defined as the sum of the weights of all its k-cliques.
Note that a 2-clique matching corresponds to a traditional matching in graphs.

A. Self-Stabilizing Protocol

Our protocol for matching nodes is inspired by the self-stabilizing distributed algorithm by Manne and Mjelde [2]. By introducing minor changes to this algorithm we create a protocol that finds a weighted k-clique matching whose total weight is off by at most a factor $k \geq 2$ in comparison to an optimal solution. In practice, we see that much better matchings are found, and most often even close to optimal.

```
1:  **loop**
2:     $C \leftarrow \{\}$
3:     **for all** $U \equiv \{u_1, \ldots, u_{k-1}\} \subseteq N(v)$ **do**
4:        **if** $attr_v(U) > attr_v(C)$ **then**
5:           $C \leftarrow U$
6:        **end if**
7:     **end for**
8:     $\mathbf{C}[v] \leftarrow C$
9:     broadcast($\mathbf{C}[v]$)
10: **end loop**
```

Figure 1. Self-stabilizing weighted k-clique matching protocol (executed by node v)

Notation: For any subset of nodes $U \subseteq V$, let $w(U)$ denote the weight of the subgraph induced by the nodes in U. In particular, $w(\{u, v\}) > 0$ denotes the weight of an edge between two adjacent nodes u and v. Moreover, let $w_v(U)$ denote the weight of a clique consisting of v and nodes in U, thus $w_v(U) \equiv w(\{v\} + U)$. The set of nodes adjacent to node v, i.e., its neighbour set, is denoted by $N(v)$.

Assumptions: We assume that each node v has a unique identifier $id(v)$ and that a total ordering is imposed on these identifiers. We also assume that the weight of each k-clique in the graph is unique[2].

Variables: As in the algorithm from [2], each node v in the network has two variables: $\mathbf{C}[v]$ denoting a set of $k-1$ neighbours of v with which v is willing to form a clique, and $\mathbf{w}[v] \equiv w_v(\mathbf{C}[v])$ denoting a weight of that clique. We consider a clique induced by nodes $v_1, v_2, \ldots, v_k$ as *matched* only if for each v_i we have that $\mathbf{C}[v_i] = \{v_1, v_2, \ldots, v_k\} - \{v_i\}$ (with $1 \leq i \leq k$). In a stable configuration each node v that is part of some clique should have all other nodes from that clique as elements of its set $\mathbf{C}[v]$. Moreover, each node v that is not part of any clique in a stable configuration should have $\mathbf{C}[v]$ empty and thus $\mathbf{w}[v] = w_v(\{\}) = 0$.

Apart from $\mathbf{C}[v]$ and $\mathbf{w}[v]$, every node also needs to know the weights of the edges between its neighbours to be able to compute the weights of the cliques it can be part of. Note that it is not necessary for a node to know the weights of all edges in the network, only the weights of the edges between its neighbours are needed. In our protocol, we assume that these weights are readily available, and abstract away from how they are obtained by each node. Note that the process of gathering information about such weights can be totally independent from the k-clique matching protocol.

Fig. 1 shows the pseudo-code of our weighted k-clique matching protocol. In an infinite loop each node in the network looks for the most attractive k-clique that it can become part of (we will formalize attractiveness shortly). In order to discover such a clique node v considers all $\binom{|N(v)|}{k-1}$

[2]Note this is easy to realize by extending each weight of a clique with a tuple of its node ids, sorted in descending order.

subsets of $k-1$ neighbouring nodes (line 3) and keeps the most attractive one.

To assess the attractiveness of a clique formed with nodes from set U, node v has to ensure that none of these nodes is currently involved in a heavier clique, because such a node would not be interested in joining a clique of a smaller weight. To this end, we call a set $U = \{u_1, \ldots, u_{k-1}\}$ of $k-1$ neighbours *proper* if and only if

$$\forall u_i \in U : w(\{v, u_1, \ldots, u_{k-1}\}) \geq \mathbf{w}[u_i]$$

and denote this fact through the predicate $\mathit{proper}(v, U)$. Only cliques constructed with proper combinations of neighbours can be considered as admissible.

Subsequently, to compare any two sets of $k-1$ neighbours, nodes follow two straightforward rules. From the perspective of node v, subset C' is better than subset C if:

- C' is *proper* and C is not, **or**
- both C' and C are *proper* and $w_v(C') > w_v(C)$.

We can express it in a more concise way by, firstly, defining the function $\mathit{attr}_v()$:

$$\mathit{attr}_v(C) = \begin{cases} w_v(C) & \text{if } \mathit{proper}(v, C) \\ 0 & \text{otherwise} \end{cases}$$

As a result, we can now check whether C' is better than C by evaluating the expression $\mathit{attr}_v(C') > \mathit{attr}_v(C)$.

By executing lines 3–7, node v chooses the heaviest admissible (most attractive) clique, setting $\mathbf{C}[v]$ and $\mathbf{w}[v]$ accordingly and broadcasting these values (line 9.) As a result, all of v's neighbours have updated information about the clique chosen by v.

In a manner similar to the one presented in [2], one can prove that the weighted k-clique matching protocol computes a solution that approximates an optimal weighted k-clique matching by at worst a factor k. Likewise, one can show that it converges in a number of rounds proportional to the number of cliques found.

B. Variable Neighbourhood Search Heuristic

To reduce the computational complexity of the task performed by each node in a single round, we incorporate a heuristic for finding the heaviest k-clique. In the protocol presented in Fig. 1 each node has to examine $\binom{|N(v)|}{k-1}$ subsets of neighbours with whom it can create a k-clique. If the number of neighbours is substantial then even for small values of k the cost of executing the protocol in its original version may be prohibitively high. Therefore, instead of checking all the possible combinations of neighbours, we propose that nodes use what is known as variable neighbourhood search (VNS) to find their k-clique. Our choice is motivated by the work of Brimberg el al. who have applied VNS to the problem of finding the heaviest k-subgraph, that is a subgraph induced on k vertices with maximal weight. In [18] they report that VNS is consistently the best over a number of other heuristics tested, including tabu search and multi-start local search.

```
function Shake(v, C, d)
  1: randomly select $U \subseteq C$ with $1 \leq |U| \leq d$
  2: randomly select $U^* \subseteq N(v) - C$ with $|U^*| = |U|$
  3: return $(C + U^* - U)$
```

Figure 2. *Shake* function randomly generates new candidate for a k-clique from $NS_d(C)$

Variable neighbourhood search is a meta-heuristic that has found many applications in solving various combinatorial and global optimization problems [19]. At the core of this method lies the interchangeable execution of two basic steps: (1) randomly changing the current solution (to avoid stagnation at a local optimum), and (2) improving the current solution by performing some form of local search. Both these steps make use of *neighbourhood structures* (hence the heuristics name) that define the sets of solutions at most d units away from some particular solution in a metric imposed on the solution space.

VNS for k-clique matching: We do not apply VNS to the weighted k-clique matching problem. Instead, we apply it to the subproblem of finding the heaviest admissible k-clique each node v can be part of. Thus, the solution space S_v consists of all $\binom{|N(v)|}{k-1}$ combinations of v's neighbours (represented by sets of $k-1$ elements each):

$$S_v = \{C | C \subseteq N(v) \text{ with } |C| = k-1\}$$

To this solution space we introduce a metric δ:

$$\delta(C, C') = |C - C'|$$

which tells us in how many elements C and C' differ. This metric is then used to construct the following neighbourhood structures NS_d:

$$NS_d(C) = \{C' | C' \in S_v \text{ with } \delta(C, C') \leq d\}$$

Note that the neighbourhood structures are nested (i.e., $NS_{d-1}(C) \subseteq NS_d(C)$) and that it makes sense only to consider $d < k$ as $NS_{k-1}(C) = NS_k(C) = \cdots$.

The *Shake* function (see pseudo-code in Fig. 2) returns a random set of $k-1$ neighbours of v that is at most d units away from a given set C. This is achieved by randomly selecting at most d elements from C and swapping these with an equal number of elements from $N(v) - C$.

The aim of the *LocalSearch* function is to improve the given solution. To this end, we need to be able to compare any two sets of $k-1$ neighbours of node v. The local search function uses the $\mathit{attr}_v()$ function to find the first improvement over a given set C among the sets that are in the first neighbourhood structure of C. This means that only the sets that differ from C by one neighbour are considered. There are at most $(k-1) \cdot (|N(v)| - k + 1)$ such sets. If no

```
function $LocalSearch(v, C)$
    for all $u \in C$ do
        for all $u^* \in N(v) - C$ do
            $C' \leftarrow C + \{u^*\} - \{u\}$
            if $attr_v(C') > attr_v(C)$ then
                return $C'$
            end if
        end for
    end for
    return $C$
```

Figure 3. *LocalSearch* finds first improvement of a clique from $NS_1(C)$

```
function $VNS\ (v, C_{opt})$
    if $C_{opt} = \{\}$ then
        randomly select $C_{opt} \subseteq N(v)$ with $|C_{opt}| = k - 1$
    end if
    while stopping condition = false do
        $d \leftarrow 1$
        while $d < k$ do
            $C \leftarrow Shake(v, C_{opt}, d)$
            $C' \leftarrow LocalSearch(v, C)$
            if $attr_v(C') > attr_v(C_{opt})$ then
                $C_{opt} \leftarrow C'$
                $d \leftarrow 1$
            else
                $d \leftarrow d + 1$
            end if
        end while
    end while
    return $C_{opt}$
```

Figure 4. VNS heuristic returns the heaviest admissible k-clique found for node v

improvement is found, C is returned as the result. Instead of using the first improvement found, we could also select the best improvement. (As the first improvement mechanism performed well in our simulations, we have not tested the best improvement mechanism.)

The VNS heuristic function is outlined in Fig. 4. In the inner loop the algorithm tries to improve the currently best solution C_{opt}. First, it randomly chooses some set C from the d-th neighbourhood of C_{opt} via the *Shake* function. Then it tries to refine C by performing a local search. If the set C' returned by *LocalSearch* is more attractive than the current best solution C_{opt} that solution is replaced and the next iteration of the loop is performed in the smallest neighbourhood structure NS_1 of the new best solution. Otherwise, when C' is not better, d is increased. Note that with the fixed set of neighbours and the information about neighbours' cliques fixed too, there is no threat of looping infinitely in the inner loop. Variable d is set to 1 only if a clique better than the currently best one is found and

```
1:  loop
2:      $C \leftarrow \{\}$
3:      if $proper(v, \mathbf{C}[v])$ then
4:          $C \leftarrow \mathbf{C}[v]$
5:      end if
6:      for all $u \in N(v)$ do
7:          if $v \in \mathbf{C}[u]$ and
            $attr_v(\mathbf{C}[u] + \{u\} - \{v\}) > attr_v(C)$ then
8:              $C \leftarrow \mathbf{C}[u] + \{u\} - \{v\}$
9:          end if
10:     end for
11:     $C' \leftarrow VNS(v, C)$
12:     if $attr_v(C') > attr_v(C)$ then
13:         $C \leftarrow C'$
14:     end if
15:     $\mathbf{C}[v] \leftarrow C$
16:     broadcast($\mathbf{C}[v]$)
17: end loop
```

Figure 5. Self-stabilizing weighted k-clique matching protocol with VNS heuristic

the number of such improvements is bounded by the finite elements in the solution space.

The inner loop is abandoned when parameter d reaches k and starts its execution anew, if possible. This searching for a better solution has to be explicitly bounded. The outer loop contains a stopping condition that limits the number of the inner loop executions. The possible stopping conditions that can be used here are: number of full inner loop executions from $d = 1$ to $k - 1$, the total CPU time, or the maximum number of iterations between two improvements. Additionally, if the executions of inner loop are too lengthy, the CPU time condition can be used to break from both inner and outer loop.

The modified k-clique matching protocol with incorporated VNS heuristic is presented in Fig. 5. Apart from executing the *VNS* function it has been extended to also check the solution found in the previous round and to consider cliques sent by neighbours that contain node v. These modifications come from considering the scope of the search performed by the VNS function in comparison to the protocol from Fig. 1. In the latter, nodes search the full solution space to find the heaviest admissible clique. There is thus no need to re-evaluate a previous solution or solutions sent by the neighbours, as they will be evaluated anyway. When the heuristic is used only part of the solution space is explored in a single round of the protocol. Thus, a very good solution from a previous round or those sent by neighbours might be forgotten/neglected causing the protocol to lose its self-stabilization property. To prevent this from happening, in lines 3–5 a node re-evaluates its solution from the previous round and in lines 6–10 it evaluates the cliques sent by neighbours. The set of $k - 1$ neighbours found in this way is thereafter fed to the *VNS* function as its starting point.

V. Experimental Results

A. Simulation Setup

In our simulations we focused on the situation in which nodes have full knowledge about the network. Thus, if there are n nodes in the network, then the size of the neighbour set $N(v)$ of each node would be equal to $n-1$. The reason for this is to control the size of the k-clique matching to be at the level of $\lfloor n/k \rfloor$ regardless of the weight assignments. However, this approach does restrict the sizes of the networks that we were able to simulate.

Apart from the fact that the list of neighbours of each node consists of all the nodes in the network, each node also has global knowledge about the weights not only between itself and its neighbours but also between each pair of its neighbours, as discussed above. This provides the node with enough information to compute the weight of any clique it can be potentially part of.

The weight between each pair of nodes is assigned uniformly at random from the interval $(0, 1)$. The weight of a clique is computed as the geometric mean[3] of the respective weights between each pair of nodes in the clique. The reasons for choosing the geometric mean over, for example, the more commonly used arithmetic mean is twofold. First, by using the geometric mean we promote cliques for which in general the weights are closer to each other. Secondly, when using the geometric mean we can use weight 0 to effectively represent the fact that two nodes cannot or should not join the same clique, thus also capturing the *absence* of an edge. Note that this also weakens our requirement (in a positive sense) that a node needs to know all weights between its neighbours: lack of knowledge can be modeled by assigning zero weight to an edge.

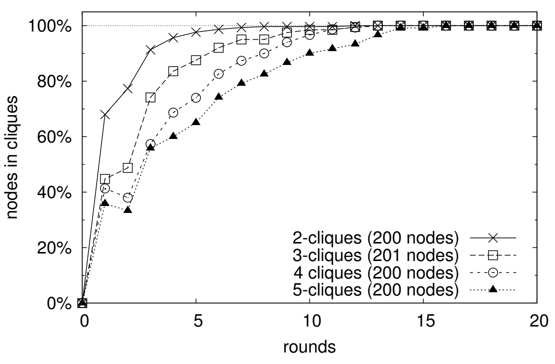
All nodes start with no initial clique chosen. They execute the protocol in a round-based fashion — in each round each node in the network executes a single loop. The order in which nodes execute their protocol in each round is random. Each simulation with distinct parameters has been executed 3 times and the results have been averaged. All simulations presented in this paper were conducted using the PeerSim simulator [20].

B. k-Clique Matching Protocol Performance

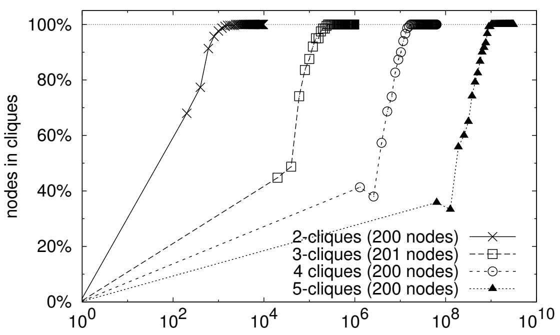
Let us first focus on the performance of the protocol without VNS. In Fig. 6a the percentage of nodes in cliques over rounds is depicted. In the network of 200 nodes the number of rounds needed for all nodes to find a matching clique increases with the value of k. Yet, even for $k = 5$ it does not exceed 20 rounds.

Nonetheless, if we present the percentage of matched cliques as a function of number of cliques evaluated by

[3]Recall that the geometric mean over a series $x_1, x_2, \ldots, x_n$ is computed as $(x_1 \cdot \cdots \cdot x_n)^{1/n}$.



(a)



(b)

Figure 6. Performance of the protocol without VNS; percentages of nodes matched into cliques for different values of k as a function of: (a) number of rounds, (b) average number of cliques evaluated by a single node since the beginning of a simulation

a single node since the start of the protocol, a completely different picture arises (see Fig. 6b, notice the logarithmic scale on the horizontal axis). This new measure can be regarded as a possible metric of the computational cost of an algorithm. The number of cliques that each node has to evaluate in a single round is equal to $\binom{|N(v)|}{k-1}$ which is proportional to n^{k-1}, hence the exponential increase in the time needed by nodes when k increases linearly. As a consequence, for even small values of k such as 4 or 5 the protocol can impose a prohibitively high computational burden on a node with substantial number of neighbours.

C. k-clique matching protocol performance with VNS

Incorporating VNS heuristic into the protocol is meant to tackle the very problem of computational load faced by nodes. Our implementation of the VNS heuristic is very basic. Moreover, while performing the simulation with VNS, we did not try to tune its parameters in any way and we arbitrarily set the *stopping condition* of the VNS to 25 full executions of the inner loop.

Fig. 7 compares the performance of the protocol without VNS to the protocol with VNS for forming 3-, 4-, and 5-cliques. As depicted in Fig. 7a and 7d, there is little difference in the effectiveness of both protocols while nodes

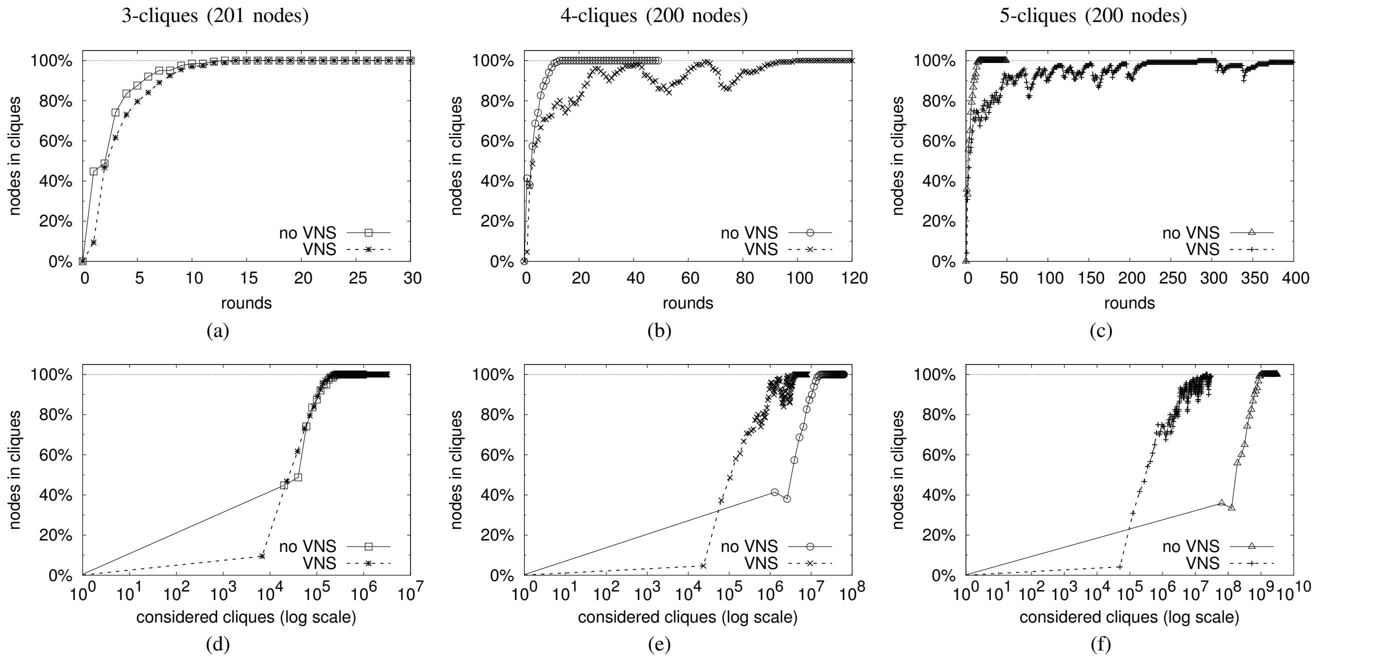


Figure 7. Percentages of nodes matched into 3-, 4-, and 5-cliques (column-wise) executing the protocol with and without VNS as a function of: (a)-(c) number of rounds, (d)-(f) number of cliques evaluated by a single node

are forming into 3-cliques. And this is true irrespective of the percentages being depicted as a function of rounds or as a function of an average number of cliques evaluated by a single node. This should not be a surprise, since for 3-cliques VNS is actually able to browse through a large part of the possible solution space while repeatedly executing the *LocalSearch* function. The situation for 4-cliques and 5-cliques is very different from the one with 3-cliques. To form into 4- or 5-cliques, nodes need substantially more rounds (note the larger range of the x-axes). Yet, in terms of the average number of cliques evaluated by a single node using the VNS is much more advantageous. For 5-cliques the gain is almost hundredfold.

When we look at the graphs for 4- and 5-cliques, we can also observe that when using VNS the percentage of nodes formed in cliques does not rise as steadily as for the protocol without VNS. With VNS a high percentage of nodes quickly compose themselves into cliques but then the percentage starts to fluctuate between 80% and 100% for some time before it finally converges to 100%. The reason for this is that many nodes initially form cliques that are not optimal (because with VNS they search only part of the possible cliques.) These cliques are soon broken by one of the nodes that finds a more attractive clique.

Fig. 8 shows that not only the percentage of nodes teamed into cliques converges to 100% when using VNS, but also that the weights of these cliques are equally high for both protocols. Fig. 8a depicts the average weight of the cliques formed as a function of rounds during a single simulation, while Fig. 8b presents the percentage of nodes composed

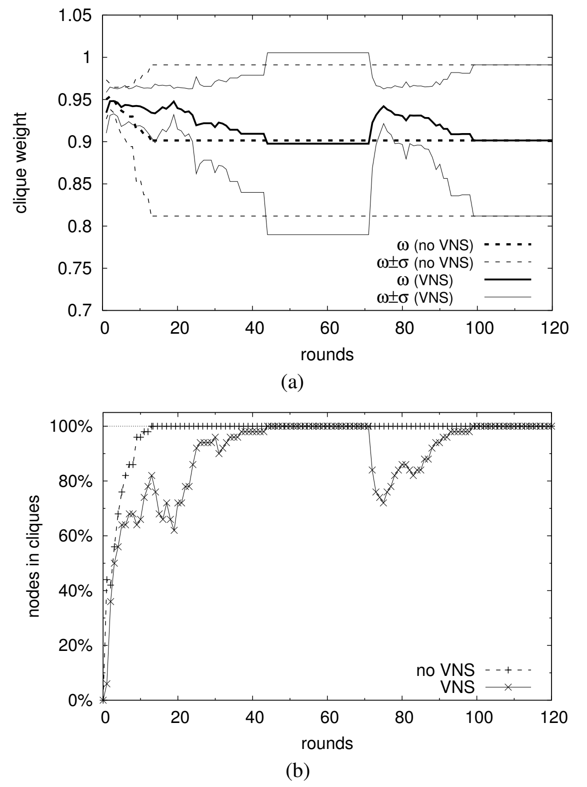


Figure 8. Performance of the protocol with and without using VNS (while forming 4-cliques in the network of 200 nodes): (a) average weight ω of a clique with standard deviations σ, (b) percentage of nodes in cliques, as functions of the number of rounds

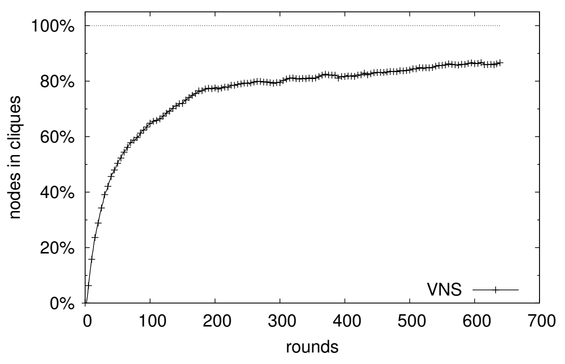


Figure 9. Performance of the VNS heuristic in the network of 10000 nodes forming into 5-cliques

into these cliques. As can be seen, when the protocol with VNS stabilizes between rounds 44 and 71 for the first time at the level of 100% of nodes matched in cliques, the average weight of these cliques is lower than the corresponding average weight obtained by nodes using protocol without VNS. Nonetheless, after some rearrangements in rounds 72–98 the percentage of nodes composed into cliques reaches 100% again and this time also the average weights of the cliques (and standard deviations) become equal for both protocols. It is therefore highly probable that the set of cliques formed by the two protocols is indeed identical.

We have also run simulations of the protocol with VNS for larger networks. Fig. 9 depicts the percentage of nodes formed into 5-cliques in the network of 10000 nodes. For this particular simulation we have set a stricter *stopping condition* that limited the time of execution of the *VNS* function. In practice, each node evaluated less than 35000 cliques per round which was just a tiny fraction of the solutions space consisting of $\binom{9999}{4} \approx 4 \cdot 10^{14}$ different cliques. In spite of that, in less than 150 rounds more than 70% nodes are composed into cliques. Another 150 rounds are needed to reach the level of 80%. In the subsequent rounds the convergence rate drops significantly. This can be attributed to the fact that, as more cliques are formed, it becomes more difficult for a node without a clique to find a set of neighbours with whom it could team up. Although at first glance, the performance of the protocol with VNS in as large network as 10000 nodes does not seem satisfactory, one must realize that in the same case the protocol without VNS would need a rough equivalent of 10^{10} rounds from Fig. 9 to execute just a single round. Moreover, the high percentage of nodes composed into cliques in this simulation was not achieved at the expense of the cliques' weights; the mean clique weight at round 500 is 0.967.

VI. Discussion

The convergence speed of our protocols, as well as their performance in general, is strongly related to the notion of self-stabilization. As defined by Dijkstra [21], the system is *self-stabilizing* if "regardless of the initial state [...] the system is guaranteed to find itself in a legitimate state after finite number of moves." As a direct consequence, self-stabilizing systems have the ability to neatly recover from transient failures such as inconsistent initializations, transmission errors, or process failures and recoveries [22]. This results in self-stabilization being a highly desirable property for protocols aimed at networks prone to frequent changes (e.g. nodes entering or leaving the network, re-wiring of the connections between nodes.)

Our protocol from Fig. 1 without the VNS is self-stabilizing. Unfortunately, by introducing the VNS heuristic into the protocol we have made it impossible to give any hard bounds on the number of moves (and ultimately, on the number of rounds) after which the network will converge into the k-clique matching of maximum cardinality in which no k nodes could form a more attractive clique. With VNS it may be possible to prove only the *probabilistic* self-stabilization of the protocol.

Nonetheless, we can still try to improve the convergence speed of the protocol with VNS in large networks. As a quick patch, we could introduce a mechanism by which nodes that are not able to find an admissible clique via the *VNS* function to try to form a clique with those neighbours which are known to also have no clique. More promising is to redefine the function of attractiveness to differentiate between non *proper* subsets of neighbours, e.g. based on the number of neighbours that are in more attractive cliques. Moreover, Brimberg at al. [18] mention that a *skewed* version of VNS (SVNS) may yield better results, specifically when k is relatively small in comparison to the number of nodes in the network. In this modification of the VNS, nodes are encouraged to explore regions of the solution space further from the current optimum. This can be achieved by allowing a node to accept results of *LocalSearch* that are slightly worse than the optimum found thus far given their distance from the optimum is sufficiently large to compensate the change.

In our simulations, we assumed that each node has a complete knowledge about all the nodes in the network and of all the weights between these nodes. Yet, in a more realistic scenario, in which nodes join and leave the network, it is impossible to make such an assumption. Besides, over time weights between particular nodes can also change in value, possibly distorting the existent formation of cliques. However, a partner discovery service, as described in Section III, can help nodes to find out about new nodes in a completely decentralized fashion. A similarly operating service can be used to gain knowledge about the weights and changes thereof. As our simulations show, the protocol converges quickly enough to account for any changes in a timely manner.

Besides working on the issues mentioned above, we intend to explore how the protocol could support formation of cliques with different number of members at the same time. It is also interesting to see how the protocol is affected when the weights are not assigned randomly but rather reflect, for example, the dependencies between real life companies, presumably resulting in values of weights being correlated. Lastly, we intend to investigate to what extent the assumption on weights symmetry can be dropped. This would allow nodes to evaluate mutual fitness differently, but in the current form of the protocol lack of weights symmetry poses the danger of deadlocks.

VII. Conclusions

In this paper, we presented a viable solution to forming k-cliques in a fully decentralized fashion, for k of limited yet reasonable size. We examined the performance of the proposed protocol through simulations and the results are encouraging. We see this as a starting-point for further research in decentralized dynamic partnership formation in the Web. The most important issues which need to be addressed are allowing differently sized partnerships to be formed and improving further the performance of the protocol.

References

[1] S. Stremersch and G. J. Tellis, "Strategic bundling of products and prices: A new synthesis for marketing," *Journal of Marketing*, vol. 66, no. 1, pp. 55–72, January 2002.

[2] F. Manne and M. Mjelde, "A self-stabilizing weighted matching algorithm," in *Stabilization, Safety, and Security of Distributed Systems*, ser. LNCS, vol. 4838/2007, 2007, pp. 383–393.

[3] G. Devlin and M. Bleackley, "Strategic alliancesguidelines for success," *Long Range Planning*, vol. 21, no. 5, pp. 18–23, 1988.

[4] C. Bronder and R. Pritzl, "Developing strategic alliances: A conceptual framework for successful co-operation," *European Management Journal*, vol. 10, no. 4, pp. 412 – 421, 1992.

[5] K. Brouthers, L. Brouthers, and T. Wilkinson, "Strategic alliances: Choose your partners," *Long Range Planning*, vol. 28, no. 3, p. 2, 1995.

[6] M. Prince and M. Davies, "Co-branding partners: What do they see in each other?" *Business Horizons*, vol. 45, no. 5, pp. 51–55, 2002.

[7] K. L. Keller, "Conceptualizing, measuring, and managing consumer-based brand equity," *Journal of Marketing*, vol. 57, no. 1, pp. 1–22, January 1993.

[8] I. E. Vermeulen, "Matchmaking in cyberspace: An application of web-based brand image measurement," in *Proceedings of the International Conference of Research in Advertising*, F. C. P. . J. Verissimo, Ed. Lisbon: Universidade de Lisboa, 2007.

[9] J. F. Moore, "Predators and prey: A new ecology of competition," *Harvard Business Review*, vol. 71, no. 3, pp. 75–86, 1993.

[10] K. Kelly, "New rules for the new economy," *Wired*, vol. 5, no. 9, 1997.

[11] D. Tapscott, D. Ticoll, and A. Lowy, *Digital capital : harnessing the power of business webs*. Boston, Mass., USA: Harvard Business School Press, 2000.

[12] E. v. Heck and P. Vervest, "Smart business networks: how the network wins," *Commun. ACM*, vol. 50, no. 6, pp. 28–37, June 2007.

[13] B. Blau, J. Krämer, T. Conte, and C. Van Dinther, "Service value networks," in *2009 IEEE Conference on Commerce and Enterprise Computing*. IEEE, 2009, pp. 194–201.

[14] M. Jelasity and O. Babaoglu, "T-man: Gossip-based overlay topology management," in *Engineering Self-Organising Systems: Third International Workshop (ESOA 2005)*, ser. LNCS, S. A. Brueckner, G. D. M. Serugendo, D. Hales, and F. Zambonelli, Eds., vol. 3910. Springer-Verlag, 2006, pp. 1–15.

[15] S. Voulgaris, M. van Steen, and K. Iwanicki, "Proactive gossip-based management of semantic overlay networks," *Concurrency and Computation: Practice and Experience*, vol. 19, no. 17, pp. 2299–2311, 2007, special Issue: Parallel and Distributed Computing (EuroPar 2005).

[16] M. Jelasity, A. Montresor, and O. Babaoglu, "T-man: Gossip-based fast overlay topology construction," *Comput. Netw.*, vol. 53, no. 13, pp. 2321–2339, 2009.

[17] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "Gossip-based peer sampling," *ACM Trans. Comput. Syst.*, vol. 25, no. 3, p. 8, 2007.

[18] J. Brimberg, N. Mladenovic, D. Urosevic, and E. Ngai, "Variable neighborhood search for the heaviest k-subgraph," *Computers & Operations Research*, vol. 36, no. 11, pp. 2885 – 2891, 2009.

[19] P. Hansen, N. Mladenović, and J. Moreno Pérez, "Variable neighbourhood search: methods and applications," *Annals of Operations Research*, vol. 175, no. 1, pp. 367–407, March 2010.

[20] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, "The Peersim simulator," `http://peersim.sf.net`.

[21] E. W. Dijkstra, "Self-stabilizing systems in spite of dsitributed control," *Communications of the ACM*, vol. 17, no. 11, pp. 643–644, November 1974.

[22] M. Schneider, "Self-stabilization," *ACM Comput. Surv.*, vol. 25, pp. 45–67, March 1993.