A Decentralized Architecture for Distributed Neighborhood Based Search

Pascal Katzenbach, Yann Lorion, Tjorben Bogon, and Ingo J. Timm

Institute of Computer Science
Goethe University Frankfurt
{katzenb,lorion,tbogon,timm}@cs.uni-frankfurt.de

Abstract. We present a decentralized self-X architecture for distributed neighborhood based search problems using an overlay network based on random graphs. This approach provides a scalable and robust architecture with low requirements for bandwidth and computational power as well as an adequate neighborhood topology, e.g. for several instances of parallel local search and distributed learning. Together with an adapted load balancing schema our architecture is self-organizing, self-healing and self-optimizing.

1 Introduction

As the number of computational resources increases faster than the computational power of single resources, in almost every discipline of computer science flexible and scalable parallel solutions are needed in order to reduce computation time. Usually, effective parallel and distributed algorithms are custom-made for each problem and no general solutions exist for bigger class of problems.

In this paper we present an architecture for the class of distributed neighborhood based search problems. The problems that we are interested in vary from parallel local search to distributed learning – basically, every search problem over a measurable search-space that can be solved efficiently by distributed search entities that are only allowed to communicate to a small given neighborhood. One possible scenario is the distribution of population based metaheuristics using island structures, i.e. encapsulated sub-populations, that are often used in evolutionary algorithms or particle swarm optimization [1].

The architecture should be self-organizing, self-healing and self-optimizing, since it should be able to deal with heterogeneous, dynamic and unreliable large-scale environments. During the last decade, peer-to-peer networks, i.e. a communication structure in which individuals interact directly without going through a centralized system or hierarchy, has proven to be an adequate solution for such requirement. In order to achieve self-organization, overlay networks are placed on top of the physical networks. Since in contrast to well-known P2P applications like file sharing we have no requirement for lookup-operations to locate explicit peers or data in the network, there is no need for structured overlay topologies, e.g. based on distributed hash tables. Instead, in order to minimize the overhead

T. Spyropoulos and K.A. Hummel (Eds.): IWSOS 2009, LNCS 5918, pp. 193–200, 2009.

for construction and maintenance, we design a new randomized overlay network that meets all requirements.

2 Fundamentals

Since the architecture, in particular the overlay network, presented in this paper is based on a construction schema for random graphs with low mixing times, we first introduce some basic definitions for graphs, markov chains, mixing times and random graphs.

An *undirected graph* G is a pair $G = (V, E)$ of a set V of n vertices or nodes and a set $E \subseteq V \times V$ of edges. The degree $\deg(v)$ of a node v is the number of edges starting (or ending) in v. A Graph G is *d-regular*, iff all nodes have degree d. A *path* is a sequence $(v_1, \dots, v_m)$ of nodes where each successive pair is connected by an edge in E. A graph G is *connected*, iff G contains a path between each pair of different nodes.

For the construction schema used in this paper, we will need to draw nodes nearly uniformly distributed out of the set of all nodes V without knowing the entire set. For that matter a random walk on the graph comes in handy. Given a pair (Ω, P) of the finite denumerable state space Ω and the stochastic $(\Omega \times \Omega)$-matrix $P = (p_{xy})$. A sequence $X_1, X_2, \dots$ of random variables is called *Markov chain*, iff it satisfies the *Markov property*

$$\Pr[X_{n+1} = y | X_n = x_n, \dots, X_1 = x_1] = \Pr[X_{n+1} = y | X_n = x_n] = p_{x_n y},$$

for all $n \geq 0$ and $x_0, \dots x_n, y \in \Omega$. By interpreting nodes as states and edges as transitions a graph can be seen as a Markov chain where $\Omega = V$ and the stochastic matrix P is denoted by $p_{ij} := 1/\deg(i)$ iff there is an edge from node i to node j. Starting with the probability vector $\mu^{(0)}$, where $\mu_i^{(0)} := \Pr[X_0 = x_i]$, the distribution at time step $t+1$ can be calculated from time step t with $\mu^{(t+1)} = \mu^{(t)} P$, hence $\mu^{(t)} = \mu^{(0)} P^t$. For $t \to \infty$ on a non-bipartite undirected graph the corresponding Markov chain has a unique stationary distribution $\pi = P\pi$, where $\pi_i = deg(i)/2|E|$ [2]. Hence when starting at an arbitrary node x and performing a random walk on a d-regular graph the distribution $P^t(x, .)$ of the node visited at time t will converge towards a uniform distribution over all nodes in V.

The *mixing time* $\tau(\epsilon)$ is the minimum number of time steps needed to almost reach the stationary distribution, i.e. to have a *maximal total variation distance* between both distributions of less than ϵ (for details see [3]). A markov chain is considered *rapid mixing* if $\tau(\epsilon) \in \mathcal{O}(\mathrm{poly}(\log(n/\epsilon)))$ holds. It can be shown, that the mixing time $\tau(\epsilon)$ is bounded by

$$\left(\frac{1}{\gamma(P)} - 1\right) \ln \frac{1}{2\epsilon} \leq \tau(\epsilon) \leq \frac{1}{\gamma(P)} \ln \left(\frac{1}{\epsilon \pi_{min}}\right) \tag{1}$$

where $\pi_{min} := \min_{x \in \Omega} \pi(x)$ and $\gamma(P)$ is the *spectral gap*, i.e. the distance between the largest and the absolute second largest eigenvalue of the stochastic matrix P[3]. Hence having a large spectral gap leads to fast mixing times.

3 Designing the Overlay Network

3.1 Requirements and Goals

The design purpose of our overlay network is to provide an architecture for distributed neighborhood based search in dynamic and unreliable large-scale environments. Since we focus on self-organizing, self-healing and self-optimization our main requirements for the overlay network are:

The communication structure should be well suited for neighborhood based search. We study distributed search entities, that are only allowed to communicate to a small neighborhood, given by the overlay network. In order to ensure fast spreading of important information the shortest way to the furthest entity should be as short as possible. Hence the graph of the overlay network should have a small diameter. A little opposing to the small diameter the number of direct neighbors of each entity, i.e. their degree in the respective graph, should be relatively small. This way even the communication costs of events that have to be spread to the entire neighborhood stay small for each entity.

The overlay network should allow dynamic joining and leaving and be robust against loss of connections and sub systems (self-organization and self-healing). One possible consequence of the requirement for robustness is that the corresponding graph should have a high degree, so that with high probability the loss of one connection has no significant effect. Another way is to let each entity have a list of backup-neighbors in case connections to neighbors fail. However losing an edge could still lead to decomposition of a connected graph to sub graphs. Hence as major requirement the graph should have a high bisection width so that a decomposition becomes highly improbable.

Operations for construction and maintenance need to be scalable to the network size (self-adaptive) and low cost intensive in computational, memory and bandwidth terms. The construction and maintenance of the overlay network should be at most logarithmic in the number of peers for the above-mentioned terms. Additionally the graph structure should be easily expandable, i.e. construction and maintenance should never necessitate restructuring and hence blocking times for communication.

Finally, the structure should be well suited for diffusion based load balancing (self-optimization). The *diffusion schema* is an iterative and local procedure for decentralized load balancing, as it only uses information of a node and its direct neighbors. We deal with a dynamic and heterogeneous environment, where each entity only knows a small neighborhood. Diffusion based load balancing is the only procedure known to the authors, that meets all requirements. In order to have a fast convergence to a balanced state the corresponding graph needs to have a high spectral gap in the diffusion matrix [4].

3.2 Model and Algorithmic Description

Some of the requirements given in the previous section result in opposed objectives. While lower boundable degrees are desirable for scalability and resource allocation reasons, they may not lead to dense graphs with small diameters. In order to reach a trade-off between these objectives, an appropriate graph structure has to be chosen.

One example for a good compromise are expander graphs. They are defined over the expansion ratio $h(G) = \min_{1 \leq |S| \leq \frac{n}{2}} \frac{|\partial S|}{|S|}$, which is a measure for density. A family of expander graphs is an infinite sequence $(G_i)_{i \in \mathbb{N}}$ of d-regular graphs with increasing sizes, if there is a $\epsilon > 0$, so that $h(G_i) \geq \epsilon$ for all $i \in \mathbb{N}$. Expander graphs as network models result in very good run-time performance in diffusion load-balancing, e.g. a static expander graph achieves in $O(1/\epsilon)$ steps a ϵ-balanced state [4]. Also, expander graphs have the useful property of logarithmic mixing-times for approximately uniform sampling.

Distributed construction of expander graphs is not a trivial task, especially in unreliable dynamic environments. One possible method are $\mathbb{H}$-Graphs [5]. A $2d$-regular random graph is constructed, which consists of d Hamilton circles. Every node knows its predecessors and successors for each circle. When a new node enters the network, for each Hamilton circle a node is chosen as insertion position via random walk sampling. Afterwards the predecessor/successor-tables of the involved nodes are updated. Hence each node has exactly $2d$ neighbors. It can be shown that these graphs are expander graphs and therefore provide uniform distributed sampling via logarithmic random walks [5]. A negative aspect for our purpose is that amongst others the failure of some connections can lead to a total restructuring process and that small graphs have to be considered separately [5].

In order to avoid these downsides we want to build an expander-similar graph with a lower bounded expansion ratio but not necessarily d-regularity and hope to keep the resulting properties of an expander graph like small diameter, a low degree and logarithmic mixing-times. This is achieved by a randomized graph construction schema where each node has the same expected degree with an as low as possible variance:

The network is modeled by a tuple (G, S, T) with an undirected graph $G = (V, E)$, where $v \in V$ are participating peer nodes and $\{v_1, v_2\} \in E$ are direct connections between peer nodes. Variable $S = (S_v)_{v \in V}$ describes the configuration of sample pools with $S_v \subseteq V \setminus v$ from which we assume at this moment, that members of S_v are (approximately) uniformly distributed in $V/\{v\}$. Symbol T is a tuple $T = \left(\overline{d}, \tau^*, \eta, d_{max}\right)$ which describes global parameters, where $\overline{d}$ indicates the target degree, τ^* indicates the minimum random walk length, η indicates the maximum sample pool size and d_{max} indicates an upper bound for degrees. Because G and S are time mutable, the precise notation would be $(G^{(t)}, S^{(t)}, T)$ with time t, but for better readability we leave (t) out.

Each peer w regulates its current degree $|Nb_w|$ to the target degree $\overline{d}$ by periodically adding or removing the corresponding number of neighbors. To add a neighbor, a node v is randomly drawn from $S_w \setminus Nb_w$ and a connection to v is established. To remove a neighbor, a node v is randomly drawn from Nb_w and the connection to v is dropped.

Figure 1 shows the basic algorithms to fill a sample pool and to sample through a random walk. FILLPOOL$_w$, which is called periodically on each peer w, is

```
FillPool$_w$()
$k \leftarrow \eta - |S_w|$
for $i \leftarrow 1, ..., k$ do
    $v \leftarrow$ Sample$_w(\tau^*)$
    if $v \notin S_w \wedge v \neq w$ then
        $S_w \leftarrow S_w \cup \{v\}$
```

```
Sample$_w(t)$
if $t \leq 0$ then return $w$
$i \leftarrow$ uniform i.i.d. over $\{1, ..., d_{max}\}$
if $i > |Nb_w|$ then return Sample$_w(t-1)$
$u \leftarrow i$-th node in $Nb_w$
return Sample$_u(t-1)$
```

Fig. 1. Pseudocode for sample pool filling and random walk sampling

responsible for filling sample pool S_w with approximately uniform distributed elements over $(V \setminus w)$. The number of added samples is denoted by the difference between target sample pool size η and the current pool size. Sampling is done via random walks of length τ^* through call of Sample$_w$. Pooled sampling brings two major advantages against on-demand sampling: Instant access on samples for faster regulation in case of differing degrees and advanced capability for network reconstruction in case of global failures (when many nodes or connections fail at once), in which on-demand sampling may not be able to reach the entire network.

Sample$_w(t)$ performs a random walk of (remaining) length t. Due to possible irregularity of network graph G (which would lead to a non-uniform stationary distribution), a technique called *max degree random walk* is used to simulate a random walk on a undirected regular graph $G' = (V, E')$ with $E' \supseteq E$: For each node $w \in V$, a number of $(d_{max} - |Nb_w|)$ self-loops are additionally added to E'. Parameter d_{max} should be with almost sure probability an upper bound for all occurring node degrees. For G' the resulting transition matrix P has the entries $P_{v,w} = 1/d_{max}$ for $w \in Nb_v$, the diagonal entries $P_{v,v} = 1 - |Nb_v|/d_{max}$ and zero entries elsewhere. This markov chain is simulated by the described local or remote recursive calls depending on a random value i in Sample$_w(t)$. In case of a connected network graph G, graph G' will also be connected and because of the added self-loops G' cannot be bipartite. Hence the markov chain on transition matrix P has a unique stationary distribution $\pi = (1/|V|, \ldots, 1/|V|)$ (see section 2). If spectral gap $\gamma(P)$ can be lower bounded, a mixing-time $\tau(\epsilon)$ in $O(\log(|V|))$ is sufficient (see inequation (1)). Resulting spectral gaps of our model are analyzed in section 4.

3.3 Load Balancing

In diffusion load balancing for a network $G = (V, E)$ with heterogeneous subsystems $v \in V$ with benchmark factors c_v, the update equations of work load W_v and work float y_{vw} between two subsystems v and w at time step t are:

$$W_v^{(t)} = W_v^{(t-1)} - \sum_{w:\{v,w\} \in E} y_{vw}^{(t)} \qquad\qquad y_{vw}^{(t)} = \alpha_{vw} \left(\frac{W_v^{(t-1)}}{c_v} - \frac{W_w^{(t-1)}}{c_w} \right) .$$

Table 1. Spectral gaps of different network sizes n and minimum random-walk lengths τ^* in 30 simulations

n	$\tau^* = 8$	$\tau^* = 12$	$\tau^* = 16$	$\tau^* = 20$	$\tau^* = 24$
500	0.0863	0.0889	0.0897	0.0889	0.0900
1000	0.0809	0.0854	0.0862	0.0866	0.0870
2000	0.0796	0.0848	0.0855	0.0858	0.0859
4000	0.0783	0.0845	0.0850	0.0852	0.0852
8000	0.0780	0.0842	0.0847	0.0849	0.0851
16000	0.0767	0.0841	0.0846	0.0848	0.0849
32000	0.0765	0.0841	0.0845	0.0847	0.0848
64000	0.0768	0.0840	0.0845	0.0847	0.0848

(a) Target degree $\overline{d} = 6$, Mean values

n	$\tau^* = 8$	$\tau^* = 12$	$\tau^* = 16$	$\tau^* = 20$	
500	0.0823	0.0846	0.0844	0.0843	0.0845
1000	0.0714	0.0829	0.0838	0.0845	0.0854
2000	0.0746	0.0822	0.0838	0.0840	0.0842
4000	0.0711	0.0836	0.0843	0.0843	0.0841
8000	0.0724	0.0833	0.0841	0.0842	0.0845
16000	0.0663	0.0834	0.0842	0.0845	0.0847
32000	0.0684	0.0836	0.0843	0.0845	0.0845
64000	0.0648	0.0819	0.0842	0.0845	0.0845

(b) Target degree $\overline{d} = 6$, Minimum values

$M = (\alpha_{vw})$ is called diffusion matrix with $0 < \alpha_{vw} < 1$ for $\{v, w\} \in E$, otherwise $\alpha_{vw} = 0$. Rate of convergence (how fast a balanced state is reached) depends on the spectral gap of M [4]. If we choose $\alpha = (1/d_{max})$, the diffusion matrix equals our transition matrix P, hence we can use the results for spectral gaps in section 4. Since distributed computation often only allows unnormalized benchmarking factors b_v, a locally Euclidean normalization $c_v = b_v/\sqrt{b_v^2 + b_w^2}$ can be applied for each pair of neighbors $(v, w) \in E$. If the distributed search problem requires steady cooperation between entities (e.g. migration in island structured metaheuristics), there is a need of constant information exchange between neighbors. Therefore a minimal float s_{min} is introduced for the computation of each outgoing float $s_{vw}^{(t)}$:

$$s_{vw}^{(t)} = s_{min} + \max\left\{0\ ,\ \alpha_{vw}^{(t)}\sqrt{b_v^2 + b_w^2}\left(\frac{W_v^{(t-1)}}{b_i} - \frac{W_w^{(t-1)}}{b_w}\right)\right\}$$

It must be pointed out that the minimal float interferes with load balancing on the way to the balanced state.

4 Evaluation

The main question of our evaluation is whether our construction schema produces graphs with expander-similar properties, i.e. leads to lower boundable expansion ratios? The verification is done by locally simulating several networks based on the described algorithm and analyzing the spectral gaps of their resulting graphs.

During simulation, new nodes are injected with rate $|V| \cdot \alpha_{join}$, existing nodes leave the network with rate $|V| \cdot \alpha_{leave}$ (rates lower than 1 are interpreted as probabilities) and existing connections fail with probability α_{fail}. In our simulations α_{join} is set to 0.4, α_{leave} is set to 0.1 and α_{fail} to 0.01. When the networks reaches a designated size, simulation is stopped and the resulting adjacency matrix is saved for eigenvalue and spectral gap computations.

Simulations with network sizes up to 64.000 nodes were performed. Table 1 shows to resulting spectral gaps. As you can see, the mean values of the simulation decrease with expanding network size. The diagonal entries in the tables suggest that spectral gaps can be lower bounded by using logarithmic minimum random-walk lengths.

Compared to structured P2P construction schemas like CHORD [6] the spectral gap of our topology does not appear to decrease monotonously with increasing number of peers. An extensive comparison of different topologies is in work.

5 Application

A first application of the presented architecture is based on distributed particle swarm optimization (PSO). PSO is a population-based metaheuristic for which one way of parallelization is the separation of the population into islands of subpopulations. Each island is computed by a different peer node in the described topology. Information exchange between islands is performed by periodic migrations of population members (particles) between adjacent peer nodes. The advantage of our network topology is reflected in an efficient spread of information in the entire network via such local exchange. Compared to other decentralized approaches for PSO (e.g. [7]) with simple topologies (e.g. circles), first results on heterogeneous networks suggests fast global information exchange with a moderate number of messages and a strong convergence to a load balanced state. Further results will be published soon.

6 Conclusion and Future Work

In this paper an innovative decentralized self-X architecture for distributed neighborhood based search problems is presented exploiting a P2P-based approach. The overlay network based on random graphs is built using approximate sampling by random walks with local sample pools. This leads on the one hand to a scalable and robust structure with low requirements for bandwidth and computational power and on the other hand provides an adequate neighborhood topology. Diffusion based load balancing ensures efficient computation on heterogeneous systems. First experiment series on a simulation of the overlay network demonstrate the properties of our architecture.

The properties of the overlay network and the corresponding random graphs are shown through simulation, but not theoretically proven yet. It would be interesting to analyze the network more theoretically, e.g. in order to achieve tighter bounds. In Addition, further simulations and evaluations are needed. First experiments with our load balancing schema show promising results but lack precise evaluation. The robustness of the overlay network has to be evaluated and test series with distributed neighborhood based search scenarios are needed. First positive results were achieved for particle swarm optimization and we are currently working on distributed learning scenarios. Another interesting point would be an empirical comparison with other random graphs as well as flooding- and especially DHT-based P2P networks for which we expect expander similar properties. Furthermore, we are currently working on a self-configuration approach for parameters $T = \left(\overline{d}, \tau^*, \eta, d_{max}\right)$. In particular the choice of a sufficient random walk length τ^* depends on the network size, for which estimation techniques are needed.

References

1. Lorion, Y., Bogon, T., Timm, I.J., Drobnik, O.: An agent based parallel particle swarm optimization - APPSO. In: Proc. of Swarm Intelligence Symposium – SIS (2009)
2. Lovász, L.: Random walks on graphs: A survey. Combinatorics, Paul Erdös is Eighty 2 (1996)
3. Levin, D.A., Peres, Y., Wilmer, E.L.: Markov Chains and Mixing Times. American Mathematical Society (2008)
4. Muthukrishnan, S., Ghosh, B., Schultz, M.H.: First- and second-order diffusive methods for rapid, coarse, distributed load balancing. Theory Comput. Syst. 31(4), 331–354 (1998)
5. Law, C., Siu, K.Y.: Distributed construction of random expander networks. In: IEEE INFOCOM 2003., vol. 3, pp. 2133–2143 (2003)
6. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM 2001: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 149–160. ACM Press, New York (2001)
7. Romero, J.F., Cotta, C.: Optimization by island-structured decentralized particle swarms. In: Proceedings of Computational Intelligence, Theory And Applications: International Conference 8th Fuzzy Days in Dortmund, Germany, September 29-October 01, 2004, Springer, Heidelberg (2005)