On Force-based Placement of Distributed Services within a Substrate Network

Laurie Lallemand* and Andreas Reifert

Institute of Communication Networks and Computer Engineering (IKR),
University of Stuttgart, Pfaffenwaldring 47, D–70569 Stuttgart, Germany
{llallem,reifert}@ikr.uni-stuttgart.de

Abstract. Network Virtualization Environments have great potential for overcoming the current ossification of the Internet, fostering innovation, and allowing several concurrent architectures to run on the same physical network. This paper presents a novel physically inspired algorithm for efficiently solving the virtual network embedding problem of placing a virtual network over a substrate network. Compared to a reference heuristic, our algorithm shows lower rejection rates and improved substrate network utilization.

1 Introduction

Internet research is at a crossroads of how to develop, deploy, test, and evaluate new architectures for next generation infrastructures that address current and future challenges. Researchers already attribute the current Internet as "ossified" [TT1] or as an "impasse" [AP1] where network architects can only suggest small deployable changes due to the Internet's multi-stakeholders principle, develop new architectures on a laboratory scale only, or must fall back to overlays with their inherent limitations.

Network Virtualization Environments (NVEs) [CB1] are new architectural approaches to overcome the ossification. A large-scale physical substrate network enables the research community to conduct experiments on a similar scale. Virtualization is the key technology, which offers and isolates the physical resources in form of slices, i. e., partitions of virtual resources. Whether an NVE is only means (base for architectural research) or end (architectural principle in itself) is undecided yet. Projects like GENI [PA1] develop such an environment with the former notion in mind, but the latter not excluded. Such an architecture would also simplify the deployment of distributed services like videoconferencing, IPTV, or content-distribution.

The survey of NVEs in [CB1] lists many more related projects and research directions. One challenge identified is the virtual network embedding problem that assigns virtual networks to slices over the substrate network. Good algorithms to solve this NP-hard problem are crucial for efficient use of the substrate's resources. There exist no recommended way on how to solve this problem

* At the time of writing, Laurie Lallemand was a student at the IKR.

best so far. Adaptation of graph theory heuristics of similar problems, application of simulated annealing, or use of integer linear solvers are current approaches. Our novel approach translates the problem into the physical domain of an n-body problem with different force types between the bodies and solve it there. The resulting algorithm scales polynomially with the problem's size and yields good results fast.

2 Problem description and related work

The *(virtual) network embedding problem* (NEP or VNEP) consists of finding a mapping of a virtual network $G_v = (V_v, E_v)$ with components V_v and connections E_v onto a substrate network $G_s = (V_s, E_s)$ with nodes V_s and links E_s. The mapping places components onto nodes ($m_V : V_v \mapsto V_s$) and implements connections through paths in the substrate network ($m_E : E_v \mapsto \mathcal{P}(E_s)$). The sources and destinations of the paths correspond with the respective locations of components' nodes. We use precomputed shortest paths and in case of several ones equal cost multi paths.

The substrate network offers resources for the virtual network. In our case, we consider offered capacity resources of nodes (cap_s) and offered bandwidth resources of links (bw_l). Components and connections have corresponding resource demands (cap_v and bw_e). The mapping must not exceed the offered resources of nodes s and links l:

$$\sum_{v:m_V(v)=s} \text{cap}_v \leq \text{cap}_s \qquad \text{and} \qquad \sum_{e:l\in m_E(e)} \text{bw}_e \leq \text{bw}_l \tag{1}$$

The virtual network is attributed with additional constraints on components and connections:

- Anchor locations $A_v \subset V_s$ of component v. Some proxy components for ingress/egress user traffic may only be positioned at certain nodes ($m_V(v) \in A_v$).
- Max. delay $\Delta t_{u,v}$ of a connection (u, v). The corresponding path must not exceed this delay ($\Delta t_{m_V(u),m_V(v)} \leq \Delta t_{u,v}$).

Figure 1 gives an example of an NEP instance. Both networks are overlaid on each other. The substrate network consists of the graph with the squared boxes. The numbered labels indicate the offered resources. The virtual network consists of the graph with the circles. The numbers indicate the resource demands. Black components are anchored to the specific nodes ($|A_v| = 1$). The remaining components can have arbitrary locations ($A_v = V_s$). The connection between components requiring capacities of 3 and 5 has an additional delay constraint.

In our approach we transform the problem into a physical model with attracting forces between components and nodes (middle) through charges, springs, and elastic bands. The components move according to the forces until they find their final locations (right).

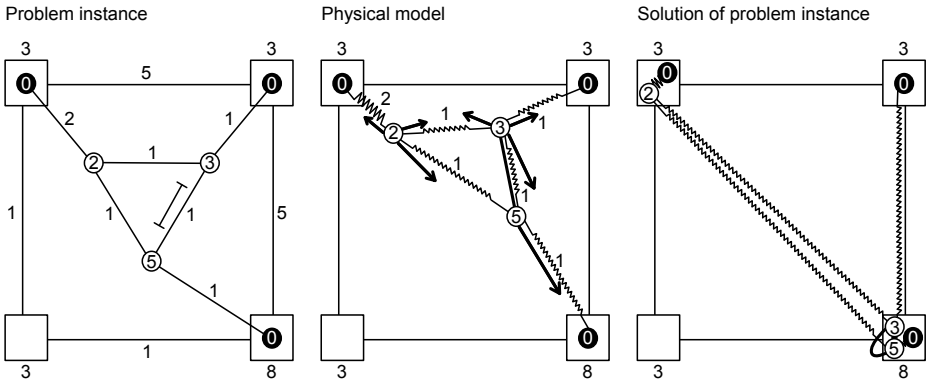

Fig. 1. Example of a network embedding problem instance (left), the corresponding physical model (middle), and its solution (right)

NEP algorithms employ various methods. [YY1] begin with a greedy component mapping and subsequently solve a multicommodity flow problem for the connections. They do not take topological information into account in the first step. [ZA1] apply a similar strategy to individual topological clusters with better results, but only consider components and connections units as resources in an unbounded environment. Others, such as [RA1], use simulated annealing. [HL1] propose an innovative distributed algorithm, but it cannot achieve competitive performance yet [CB1].

Very recent works take an integrated approach. [LK1] adapt an algorithm for finding subgraphs in the substrate topology. [CR1] model the problem as an integer linear program and approximate the solution with LP-relaxation. We take an integrated approach as well, which adapts a physical force-based model.

Graph drawing algorithms make successfully use of physical force-based models for finding layouts that intuitively visualize explicit or inherent graph properties like edge lengths, edge weights or vertex distances in the best possible way. The model in [KK1] replaces the graph with a rings-and-spring system and finds a configuration with minimal potential energy. The model in [FR1] applies attractive forces on edges' endpoints and repulsive forces on each vertex pair on a system of particles representing nodes. Our model combines both approaches.

3 Physical model

We replace vertices with charged particles and connections with springs and elastic bands. Our model is general enough to host additional forces if further constraints arise.

3.1 Particles

For each node $s \in V_s$ with offered capacity cap_s, we introduce a negatively-charged particle $p^{-\text{cap}_s}$ (anion). For each component $v \in V_v$ with capacity de-

mand cap_v, we introduce a positively-charged particle $p^{+\mathrm{cap}_v}$ (cation). We denote the set of anions and cations as P^- and P^+. Particles are arranged within an n-dimensional Euclidean space. Currently, we use two dimensions, but our method applies to arbitrary ones. The space has an Euclidean norm $\|\cdot\|$ and defines a distance $\mathrm{d}_{p,q}$ between two particles p and q according to their positions. We overlay the space with a grid of squared cells with identical cell dimensions. The grid enables efficient lookup of short-distance neighboring particles.

The anions occupy the fixed node positions in the space. Initially, all cations with only one anchor location are bound to their anchor anion and will stay fixed there. The remaining cations are free, randomly positioned, and mobile. We denote the sets of bound and free cations with P_{bound} and P_{free} .

In this physical system, charges express the capacity resources. A $p^{+\mathrm{cap}_v}$ cation may bind with any sufficiently charged anion $q^{-\mathrm{cap}_s}$ through an reduction-oxidation reaction if $\mathrm{cap}_v \leq \mathrm{cap}_s$. After the reaction, the charge of the anion is reduced by cap_v. We will later specify the exact conditions when a cation can bind (see subsection 4.2).

$$q^{-\mathrm{cap}_s} + p^{+\mathrm{cap}_v} \;\rightarrow\; q^{-(\mathrm{cap}_s - \mathrm{cap}_v)} + p \tag{2}$$

3.2 Springs and elastic bands

For each connection $e \in E_v$ with positive bandwidth demand bw_e, we introduce a spring between the corresponding endpoints' cations. The static equilibrium of such a spring is the unextended state. Its spring constant is the respective bandwidth demand. Thus, components with a large bandwidth demand come closer: the connection consumes less bandwidth in the substrate network.

On a real world partially-meshed network, the Euclidean distance between two nodes is correlated with the shortest path delay between those two nodes. Using statistical estimation, we can thus model a maximum allowed Euclidean distance between two delay-restricted components p and q with a linear function on $\Delta t_{p,q}$. We introduce an elastic band between the respective cations, of length of the maximal allowed distance between the particles, denoted $\mathrm{d}_{\Delta t_{p,q}}$. The band is unstressed when the constraint is not violated. When it is violated, the band is stressed with global spring constant c.

3.3 Forces and potential energy

The model elements "charged particles", "springs", and "elastic bands" exert forces on the cations components in P_{free}. We can consider each individual force between each individual pair of particles. We have three types of forces $\mathrm{FT} = \{\mathrm{cap}, \mathrm{bw}, \mathrm{delay}\}$ referred to as capacity-based type, bandwidth-based type, and delay-based type.

In the following, we consider one special free particle $p \in P_{\mathrm{free}}$. We denote the force of type t that another particle q exerts on p with $\boldsymbol{f}_{p,q}^{(t)}$. The force in the electrical central field of q points to q; the force of the spring or elastic band points to q, too. The resulting force on a particle is:

$$\boldsymbol{f}_p^{(t)} = \sum_{q \in N_p^{(t)}} \boldsymbol{f}_{p,q}^{(t)} \qquad , \qquad \boldsymbol{F}_p = \sum_{t \in \mathrm{FT}} \boldsymbol{f}_p^{(t)} \tag{3}$$

The magnitude $\|\boldsymbol{f}_{p,q}^{(t)}\|$ depends on the force type, the particle distance, the charges, and the spring or elastic band constants. Also, not all particle pairs are relevant. Bandwidth and delay forces only apply between spring and elastic-band-connected cations. For capacity-based forces we only concentrate on free cation-anion pairs with the possibility to bind. Thus, each type defines for each particle p a neighborhood $N_p^{(t)} \subset P$ of relevant counterparts q. Table 1 gives the magnitudes and neighborhoods for the different force types. w_{cap}, w_{bw} and w_{delay} serve to weight the relative effect of forces. Equalling w_{cap} to the area of the grid and the other weights to unity gives good results.

Capacity-based forces make compatible particles to be attracted to each other. The formula is inspired from Coulomb's Law: it is proportional to the product of charges between the two particles and diminishes quadratically with their distance. Thus, components are drawn to high capacity nodes, but only within a certain radius. Components with larger demand reach them faster. As there is no reason for two components to be spread apart, this capacity force is never repulsive: $N_p^{(\mathrm{cap})}$ does not contain other cations. The averages of all negative and positive charges normalize the force so that its magnitude does not depend on the problem instance.

Bandwidth-based forces are spring-like interactions between any two cations that share a connection. The formula is Hook's Law, for a spring whose static equilibrium is to be completely retracted and whose spring constant is the normalized bandwidth of the shared connection.

Delay-based forces are the interactions exerted by elastic bands which gets stretched at the maximum delay distance between any two particles: it is zero until the delay distance is exceeded; then, it acts as a spring with global spring constant c, whose static equilibrium is obtained at the delay distance.

Table 1 also gives the partial potential energies that constitute the system's potential energy differentiated by force type. The potential energy is a relative measure of the energy stored within the system. The difference in potential energies between two configurations of a system indicates the work necessary to get from the first configuration to the second one. If the potential energy is minimal and the particles are not moving the system is stable.

Table 1. Forces magnitudes, neighborhoods and associated potential energies

$t \in \mathrm{FT}$	$\|\boldsymbol{f}_{p,q}^{(t)}\|$	$N_p^{(t)}$	$u_{p,q}^{(t)}$
cap	$w_{\mathrm{cap}} \cdot \frac{\mathrm{cap}_p \mathrm{cap}_q}{\mathrm{cap}_{\mathrm{avg}+} \mathrm{cap}_{\mathrm{avg}-}} \cdot \frac{1}{\mathrm{d}_{p,q}^2}$	$\left\{q \in A_p \mid \mathrm{cap}_q \geq \mathrm{cap}_p\right\}$	$-\mathrm{d}_{p,q} \cdot \|\boldsymbol{f}_{p,q}^{(\mathrm{cap})}\|$
bw	$w_{\mathrm{bw}} \cdot \frac{\mathrm{bw}_e}{\mathrm{bw}_{\mathrm{avg}}} \cdot \mathrm{d}_{p,q}$	$\left\{q \in P^+ \mid (p,q) \in E_v\right\}$	$\frac{1}{2} \cdot \mathrm{d}_{p,q} \cdot \|\boldsymbol{f}_{p,q}^{(\mathrm{bw})}\|$
delay	$w_{\mathrm{delay}} \cdot c \cdot \left(\mathrm{d}_{p,q} - \mathrm{d}_{\Delta t_{p,q}}\right)$	$\left\{q \in P^+ \mid \mathrm{d}_{p,q} \geq \mathrm{d}_{\Delta t_{p,q}}\right\}$	$\frac{1}{2} \cdot \left(\mathrm{d}_{p,q} - \mathrm{d}_{\Delta t_{p,q}}\right) \cdot \|\boldsymbol{f}_{p,q}^{(\mathrm{delay})}\|$

It derives from the positions of the physical bodies and forces that are applying on them. In our case, it is calculated as the sum of the individual partial potential energies that are defined between pairs of particles exerting forces on each other. Those individual potential energies only depend on the magnitude of forces and distance between the cation and the force-inducing particles.

$$u_p^{(t)} = \sum_{q \in N_p^{(t)}} u_{p,q}^{(t)} \quad , \quad U_p = \sum_{t \in \mathrm{FT}} u_p^{(t)} \quad , \quad U = \sum_{p \in P^+} U_p \tag{4}$$

4 Force-based placement algorithm

The forces determine the dynamic behavior of our system of particles over time. We consider individual discrete time steps. At each step we calculate the movement of each particle according to the forces, determine the new positions and reset all velocities to zero.

4.1 Displacement

The linear equations of [KK1] does not apply to our model because we have other types of forces than spring-based forces. [FR1] describes a simplified model of displacement with one attractive and one repulsive force types. The displacement step is proportional to the resulting force applying on the particle and is limited by a temperature. The value of the temperature acts as a cut-off for the displacement step of each particle. A geometrically decreasing temperature series $\mathrm{T}^{(n)}$ leads to the final configuration, avoiding infinite oscillations.

We take a similar approach to displace free cations, except that our temperature is a scaling factor instead of a cut-off value and that we also use it to avoid infinite displacement due to infinite-magnitude forces. $\Delta \boldsymbol{x}_p^{(n)}$ is the normalized displacement of p at time step n:

$$\Delta \boldsymbol{x}_p^{(n)} = \frac{\mathrm{T}^{(n)}}{|P^+|} \cdot \frac{\boldsymbol{F}_p}{\max_{q \in P} \|\boldsymbol{F}_q\|} \tag{5}$$

If one delay constraint is too much violated, i.e. if $\|\boldsymbol{f}_q^{(\mathrm{delay})}\|$ exceeds the cell length of the grid, then only delay forces apply in the calculation of the next displacement steps of all particles:

$$\Delta \boldsymbol{x}_p^{(n)} = \frac{\mathrm{T}^{(n)}}{|P^+|} \cdot \frac{\boldsymbol{f}_p^{(\mathrm{delay})}}{\max_{q \in P} \|\boldsymbol{f}_q^{(\mathrm{delay})}\|} \tag{6}$$

4.2 Reaction conditions

At any time, any couple of particles that meet the requirements of Tab. 2 can react according to (2). This determines the set $P_{\mathrm{react}}^{(n)}$ of all free cation-anion

Table 2. Reaction conditions

Condition	Description
Charge compatibility	$\mathrm{cap}_v \geq \mathrm{cap}_s$
Anchoring	$s \in A_v$
No delay violation	$\|\boldsymbol{f}_p^{(\mathrm{delay})}\| = 0$
Enough underlying bandwidth	Connections between all components that correspond to a cation in $P_{\mathrm{bound}} \cup \{p\}$ must be supported on links.
Proximity	$\mathrm{d}_{p,q} \leq$ cellsize
Stability	Displacement due to non-electrical forces must be small (free cation slows down near a good position).

pairs that can react at time step n. Each reaction actually places the component of the cation on the node of the anion. The electron transfer corresponds to a reservation of node capacity.

4.3 Stages

Due to the dominance of charge-based forces in the short range, activating all forces at once leads to mediocre results. Thus, our algorithm consists of two stages during which free particles move into equilibrium through Alg. 1. The only difference between the two stages are the forces activated, the temperature cooling schedule, and that particles can only bind in the second stage.

The first stage starts with all free particles randomly placed. Only bandwidth and delay forces are activated. Bandwidth forces leads to the equilibrium of a spring system. Delay forces ensure that no delay requirement is broken. This stage is cooled down with positive rate $\alpha_{1,\mathrm{cool}} < 1$ by means of a simple geometrically decreasing temperature ($\mathrm{T}^{(n+1)} = \alpha_{1,\mathrm{cool}}\mathrm{T}^{(n)}$).

The second stage starts from the end position of the first stage. All forces are activated and cations/components are drawn to anions/nodes. At the end of each time step, the couple of particles that meet the reaction conditions of Tab. 2 and whose particles are closer to each other react. Because the electrical state of anions change, cations have to reorganize after each reaction. Hence we need a more aggressive temperature cooling schedule that cools down very fast (rate $\alpha_{2,\mathrm{cool}} < \alpha_{1,\mathrm{cool}}$) and reheats with factor $\alpha_{2,\mathrm{heat}} > 1$ after one cation has bound or no cation has found any suitable anion during the current cooling round:

$$\mathrm{T}^{(n+1)} = \begin{cases} \alpha_{2,\mathrm{cool}}\mathrm{T}^{(n)} & \text{if } \mathrm{T}^{(n)} > \mathrm{thresh} \wedge |P_{\mathrm{free}}^{(n+1)}| \geq |P_{\mathrm{free}}^{(n)}| \\ \alpha_{2,\mathrm{heat}}\mathrm{T}^{(n)} & \text{otherwise} \end{cases} \tag{7}$$

At the end of the second stage, remaining cations, if any, are classified by charge in decreasing order and greedily forced to react with the nearest anion for which all reactions conditions except the last two are fulfilled. If some cations are left after this process, the placement fails.

Algorithm 1 Displacement algorithm for both stages

Input: P, the set of all particles
Input: $\boldsymbol{x}_p^{(0)}$, the initial position of particles
Input: $\mathrm{T}^{(n)}$, a series of temperatures
Input: $C^{(n)}$, a series of stopping conditions

$n \leftarrow 0$
repeat
  $n \leftarrow n + 1$
  for $p \in P$ **do** {calculate forces and potential energy}
    for $t \in \mathrm{FT}$ **do**
      Calculate $\boldsymbol{f}_p^{(t)}$ according to (3)
    end for
    Calculate $u_p^{(t)}$ according to (4)
  end for
  if not $C^{(n)}$ is met **then** {displace particles}
    for $p \in P_{\mathrm{free}}^{(n)}$ **do**
      Calculate $\Delta\boldsymbol{x}_p^{(n)}$ according to (5) or (6)
      $\boldsymbol{x}_p^{(n)} \leftarrow \boldsymbol{x}_p^{(n-1)} + \Delta\boldsymbol{x}_p^{(n)}$
    end for
  end if
  if $P_{\mathrm{react}}^{(n)} \neq \emptyset$ **then** {bind particles}
    Choose random $(p, q) \in \arg\min_{(p,q) \in P_{\mathrm{react}}^{(n)}} \{\mathrm{d}_{p,q}\}$
    Bind p to q according to (2)
  end if
until $C^{(n)}$ is met

Each stage must stop once all forces are equilibrated. When this happens, the system is near a stable state of low potential energy, which means that the current position of particles is a satisfying compromise regarding initial requirements. In [KK1], the algorithm stops when the potential energy is under a not further specified threshold. In [FR1], it stops after an experimentally found fixed number of moves. Unfortunately, both thresholds are problem-specific in our model.

Thus, we stop a stage when the magnitude of the averaged resulting force for the last moves is low for most particles. This condition has the advantage of treating particles that are not moving anymore and particles that are infinitely oscillating equally. Sometimes, particles have not enough time to arrange themselves so that this first condition is met. To avoid loosing time while the system does not evolve anymore, we also end a stage when the average variation of potential energy for the recent last moves is low enough. $C^{(n)}$ aggregates all those conditions into one stopping criterion.

5 Evaluation

Our evaluations compare the performances of our *force-based* algorithm (FB) in different scenarios with the greedy *best-node-first* heuristic (BNF) of [YY1]

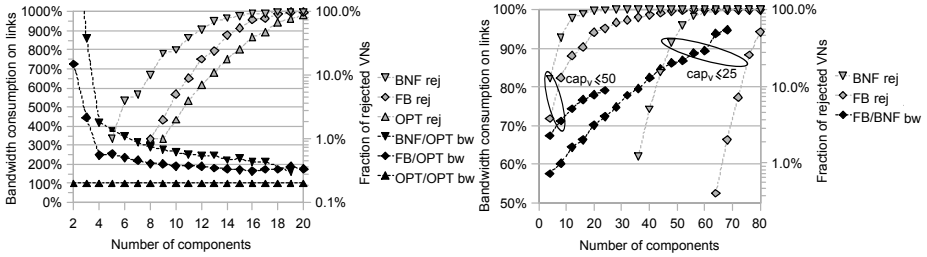

Fig. 2. Comparison of rejection rates (grey) and bandwidth consumption (black) of our force-based algorithm (FB) with a best-node-first heuristic (BNF) and optimal solutions (OPT)

and, for small instances only, with the *optimal* solution (OPT). For OPT we developed a mixed integer linear program formulation and solved it with SCIP [Ko1], with almost prohibitive running times for the generated instances already.

5.1 Evaluation scenarios

We based our evaluations on Monte-Carlo simulations with 100 or 500 problem instances per diagram value. Our networks are undirected graphs.

For each problem instance, we generate random substrate network graphs G_s of 10 or 50 nodes according to the Waxman model [Wa1] with parameters $\alpha = 0.1$ and $\beta = 0.9$. Node capacities are uniformly distributed integers in the interval $[10, 50]$. Links have infinite available bandwidth. Our virtual network G_v is a random graph with a variable number of non-anchored components from 2 to 80. It is generated according to the Erdös-Rényi model [ER1] with connection probability between two components of 25%. Component capacities are uniformly distributed integers in the intervals $[10, 25]$ or $[10, 50]$. Each of those non-anchored components has a 30% chance of being connected to a proxy, modeled by an additional component anchored to a single node.

5.2 Comparison

Both diagrams of Fig. 2 display the results of our experiments, in particular rejection rates, as light grey markers with associated logarithmic axis on the right, and bandwidth consumption, as blacks markers with associated axis on the left. OPT results appear as upwards triangles, FB results as diamonds and BNF as downwards triangles.

Our first scenario concerns small problem instances (Fig. 2, left). We independently compare FB and BNF with the optimal solution and thus only can use 100 problem instances per diagram value. Independent means that for bandwidth consumption we consider non-rejected instances of any algorithm irrespective of any other. In our case, for FB we always consider at least the same instances as

for BNF. Each substrate network has 10 nodes. There are 2 to 20 components with required capacities in the interval between 10 and 25.

With increasing virtual network size, we observe a growth in the rejection rates, the rejection rate of FB being much lower than that of BNF. The rejection rate of FB is actually closer to OPT than to BNF. Rejection is low – below 10% – for at most 8 components in BNF, for at most 11 components in FB, and for at most 12 components in OPT. In this area, FB's rejection rate is below one fifth of BNF's one.

With increasing virtual network size, the averaged normalized bandwidth consumption with respect to OPT decreases. The main contributing factor for this decrease is the reduced probability of finding really bad placements. For this reason, meaningful bandwidth consumption results concerns only areas with a low rejection rate. The bandwidth consumption of FB is 2 to 4 times higher than the optimum value, and at most 75% that of BNF in independent comparison. Further investigation revealed that if we compare only problem instances that have been accepted by both BNF and FB directly, FB bandwidth consumption even drops below 40% of that of BNF.

Our second scenario deals with large problem instances and compares FB and BNF directly (Fig. 2, right). We could not run OPT anymore in reasonable time. We use 500 problem instances per diagram value. Each substrate network has 50 nodes. There are 4 to 80 components with required capacities between 10 and cap_v with $\text{cap}_v = 25$ or 50.

Again, FB rejects significantly less problems than BNF. For the small capacities configuration, the rejection rate of FB is below 10% as long as the number of components is below 72. BNF already exceed this threshold with more than 40. For the high capacity configuration, the limit is 4 components for FB, while BNF always rejects more than 10% problems. Concerning bandwidth consumption, the bandwidth consumption of FB is always less than 80% that of BNF in low rejection areas for both components' capacities configurations.

6 Conclusion

Network Virtualization Environments are a promising approach for large-scale Internet network architecture experimentation and can also form the base of a future Internet architecture on its own. One challenge there is solving the NP-hard network-embedding problem. We transposed such problem instances into a physical system with charged particles, springs and elastic bands, and designed a force-based placement algorithm inspired from graph layout algorithms. However, our model introduces more attractive force types. Therefore, our algorithm runs in two stages instead of just one. Our approach currently supports capacity and bandwidth offers/demands as well as delay constraints. It can be easily extended through additional force types.

Monte-Carlo simulations indicate that our algorithm improves the rejection rate and the bandwidth consumption over a BNF heuristic, while still being scalable to large problem instances in terms of running time. In particular, our

force-based algorithm rejects less instances of hard problems than the heuristic, and significantly less instances of average difficulty. In the latter case, average bandwidth consumption obtained by the force-based algorithm is at least 20% lower than the independent results of the BNF heuristic. In direct comparison, it is even at least 60% lower. In small simulation scenarios where we could obtain the optimum solutions, our algorithm consumes twice the optimal bandwidth.

References

[AP1] Anderson, T., Peterson, L., Shenker, S., Turner, J.: Overcoming the Internet Impasse through Virtualization. IEEE Computer, vol. 38, nb. 4, 34–41. 2005.

[CB1] Chowdhury, N.M.M.K., Boutaba, R.: A survey of network virtualization. Computer Networks. 2009.

[CR1] Chowdhury, N.M.M.K., Rahman, M.R., Boutaba, R.: Virtual Network Embedding with Coordinated Node and Link Mapping. 2009.

[ER1] , Erdös, P., Rényi, A.: On Random Graphs. Publicationes Mathematicae Debrecen, vol. 6, 290–297. 1959.

[FR1] Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. Softw. Pract. Exper., vol. 21, nb. 11, 1129–1164. 1991.

[HL1] Houidi, I., Louati, W., Zeghlache, D.: A Distributed Virtual Network Mapping Algorithm. 2008.

[KK1] Kamada, T., Kawai, S.: An Algorithm for Drawing General Undirected Graphs. Inf. Process. Lett., April, nb. 1, 7–15. 1989.

[Ko1] Konrad-Zuse-Zentrum für Informationstechnik Berlin: SCIP—Solving Constraint Integer Programs. http://scip.zib.de/.

[LK1] Lischka, J., Karl, H.: A Virtual Network Mapping Algorithm based on Subgraph Isomorphism Detection. 2009.

[LT1] Lu, J., Turner, J.: Efficient Mapping of Virtual Networks onto a Shared Substrate. 2006.

[PA1] Peterson, L., Anderson, T., Blumenthal, D., Casey, D., Clark, D., Estrin, D., Evans, J., Raychaudhuri, D., Reiter, M., Rexford, J., Shenker, S., Wroclawski, J.: GENI Design Principles. IEEE Computer, vol. 39, nb. 9, 102–105. 2006.

[RA1] Ricci, R., Alfeld, C., Lepreau, J.: A Solver for the Network Testbed Mapping Problem. ACM SIGCOMM Computer Communications Review, vol. 33, nb. 2, 65–81. 2003.

[TT1] Turner, J. S., Taylor, D. E.: Diversifying the Internet. IEEE Global Telecommunications Conference (GLOBECOM '05), 28 nov.–2 dec., 755–760. 2005.

[Wa1] , Waxman, B.M.: Routing of Multipoint Connections. IEEE Journal on Selected Areas in Communications, vol. 6, nb. 9, 1617–1622. 1988.

[YY1] Yu, M., Yi, Y.,Rexford, J.,Chiang, M.: Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration. ACM SIGCOMM Computer Communication Review, vol. 38, nb. 2, 17–29. 2008.

[ZA1] Zhu, Y., Ammar, M.H.: Algorithms for Assigning Substrate Network Resources to Virtual Network Components. INFOCOM 25th IEEE International Conference on Computer Communications, Proceedings, 1–12. 2006.