Empowering Self-Driving Networks

Patrick Kalmbach[1] Johannes Zerwas[1] Péter Babarczi[1] Andreas Blenk[1]
Wolfgang Kellerer[1] Stefan Schmid[2]
[1] Technical University of Munich, Germany [2]University of Vienna, Austria

ABSTRACT

As emerging network technologies and softwareization render networks more flexible, the question arises of how to exploit these flexibilities for optimization. Given the complexity of the involved network protocols and the context in which networks are operating, such optimizations are increasingly difficult to perform. An interesting vision in this regard are "self-driving" networks: networks which measure, analyze and control themselves in an automated manner, reacting to changes in the environment (e.g., demand), while exploiting existing flexibilities to optimize themselves.

A fundamental challenge faced by any (self-)optimizing network concerns the limited knowledge about future changes in the demand and environment in which the network is operating. Indeed, given that reconfigurations entail resource costs and may take time, an "optimal" network configuration for the current demand and environment may not necessarily be optimal also in the near future. Thus, it is desirable that (self-)optimizations also *prepare* the network for possibly unexpected events.

This paper makes the case for *empowering* self-driving networks: empowerment is an information-centric measure which accounts for how "prepared" a network is and how much flexibility is preserved over time. While empowerment has been successfully employed in other domains such as robotics, we are not aware of any applications in networking. As a case study for the use of empowerment in networks, we consider self-driving networks offering topological flexibilities, i.e., reconfigurable edges.

KEYWORDS

Self-driving Networks, Optimization, Network Intelligence

1 INTRODUCTION

The increasing complexity of communication networks, their continuously changing requirements (e.g., in terms of demands and workloads), and their complex objective functions, render their management *in real time* almost impossible for human operators with today's tools. Indeed, there is an increasing consensus that network operations should be supported by data-driven, machine-learning-based models revolving around high-level goals and a holistic view of the underlying network [5]. An increased automation bears the potential to not only simplify network operations but also enable more fine-grained optimizations, fully leveraging the available network data rather than relying on predefined models. A vision emerges of fully self-driving networks which measure, analyze and control themselves continuously.

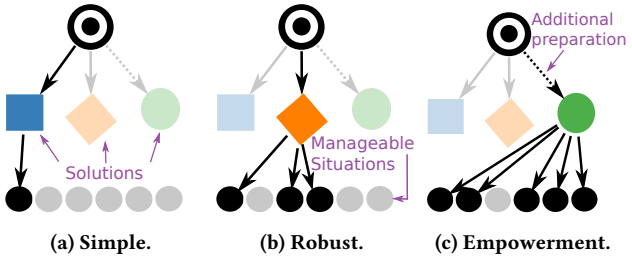


Figure 1: Comparison of optimization approaches with respect to the number of situations they can handle. Simple is the optimization for specific inputs. Robust corresponds to robust or stochastic optimization. Empowerment is the envisioned optimization through an autonomous agent.

Another major motivation for self-driving networks comes from the increasing flexibilities offered in modern communication networks. Indeed, networks have become more and more software-defined and reconfigurable. However, exploiting such flexibilities is non-trivial. Even with state-of-the-art machine learning algorithms and utilizing all the available information about the current network state and demands, any algorithm optimizing the network is faced with the challenge that there is an inherent uncertainty regarding the future: there may be an unexpected spike in the demand, an edge failure or any other unexpected behavior of a component internal or external to the network. Moreover, reconfigurations typically come at a cost, e.g., wrt. resource consumption or time (and hence service disruption), even for the most flexible and reconfigurable self-driving network. Accordingly, in addition to adjusting optimally to the *current* situation, a self-driving network should also be *prepared* for the requirements that may come up in the near future. That is, networks should be optimized *robustly*, accounting for present and possible future demands.

This paper initiates the discussion of how to optimize (self-driving) communication networks, and keep open as many options as possible. Indeed, while the need for this additional "preparedness" may seem intuitive and known on an anecdotal level, little is known on how to actually prepare a self-driving network and enhance it with the intelligence it will need to perform well also in the future. Worse, today, we even lack good definitions and models to capture such properties.

In order to fill this gap, in this paper we establish a connection to an intriguing notion of *preparedness*, so far only used and successfully applied in other domains, such as robotics. In particular, we make the case for applying the framework of *empowerment* [13]

SelfDN 2018, August 24, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5914-6/18/08…$15.00
https://doi.org/10.1145/3229584.3229587

to communication networks. Empowerment is an information-theoretic measure to quantify the influence of an "agent" (or actor) on its environment.

Our Contributions. In this position paper, we make the case for introducing empowerment in future self-driving networks: networks for which specifying complex objective functions is undesirable and cumbersome and which should not only optimize for the current demand but also be prepared for upcoming changes. We describe the challenges involved in enhancing communication networks with empowerment and present a concrete proposal based on a basic case study revolving around emerging reconfigurable network designs. Our preliminary experiments demonstrate that using empowerment as a driver for action selection, high reconfiguration costs and potentially harmful situations may be avoided.

Novelty and Related Work. Our approach radically differs from classical approaches which use utility functions to guide optimizations and suffer from the drawback of having to design and tweak the functions on a case by case basis. The concept of empowerment itself was introduced in [13], and studied by many authors subsequently, e.g., [1, 18, 20]. Successful examples of empowerment to control tasks are, e.g., the inverted pendulum [9], the control of a bipedal robot [10] or the control of a human follower [15]. Empowerment can be used as a task-independent, intrinsic motivation to restructure the environment: in our case *the network*. In general, the need and benefits of being "flexible" [11] or "prepared" is an ever-green topic in networking research, and one of the goals in Robust Optimization (RO) and Stochastic Optimization (SO) [3]. RO and SO can handle parameter uncertainty and are thus not sensitive to perturbations in the input parameters, as opposed to deterministic optimization methods. One prominent example of RO in networking is oblivious routing [2], where routes need to be defined without full knowledge of the traffic matrix, or resilient routing where edges themselves are uncertain [14, 21].

The benefit we see in empowerment compared to RO and SO is the ability of agents to learn empowerment from data, and thus do not need a comprehensive mathematical model of the system, i.e., the network. The idea is to enable the network to learn controlling itself *through experience*. We see empowerment as a possibility to drive a system that is too complex to model (as is the case for today's networks [5]) towards a state of "preparedness", maximizing situations the system can cope with.

To the best of our knowledge, we are the first to consider empowerment in the context of communication networks, motivated by the advent of self-driving networks.

2 EMPOWERMENT IN NETWORKS

In this section, we introduce the concept of *empowerment*, discuss its application in communication networks, and initiate the discussion of a case study of a reconfigurable network which needs to serve routing requests.

2.1 General Concepts

Empowerment is a concept from information theory. It is motivated by the observation that living organisms strive for states that give them maximum control or influence over their environment: Everything else being equal, states are preferable which (1) keep as many options open as possible, or (2) in which actions have the largest influence on the direct environment [13, 19]. The concept of empowerment is an attempt to formalize and quantify the influence an agent has on its environment. This contrasts with current approaches in communication systems, such as RO, SO and simple optimization (which does not include parameter uncertainty).

Fig. 1 illustrates this difference abstractly. The top layer represents an initial system state that should be adapted. The middle layer corresponds to solutions, and the bottom layer to situations that can be addressed by the system, e.g., a different demand as expected or hardware failures. Fig. 1a shows a simple optimization with specified input parameters. While the solution might be optimal for this specific input, it might restrict the number of situations that can be handled. RO and SO address this problem and model uncertainty in the input. Therefore, more situations can potentially be handled, as illustrated in Fig. 1b. However, the required mathematical model necessarily makes assumptions and abstractions of the system, which might impact the number of manageable situations [3]. Empowerment, in contrast, is based on data obtained from the system utilizing it to restructure the system towards states that maximize the number of options, leading to a potentially larger number of manageable situations, as illustrated in Fig. 1c.

A key aspect of empowerment is the agent's *embodiment*: The sensomotoric capabilities of an agent [18]. The interplay of agent and environment can be represented as a perception-action loop, where the agent influences the environment through its actuators, and receives a perception of the resulting environmental state through its sensor [18]. In networking, actuators could, e.g., change the topology or adapt routing weights, while sensors could be Quality of Service measures or the number of routed flows.

Sensors and actuators are thereby connected through the environment. An agent chooses an action for the next time step based on the sensory information of the current time step. The action influences the state of the environment, which in turn affects the sensor input. The cycle then repeats itself. This perception-action loop can be modeled formally through Random Variables (RVs):

- the sensor S taking values $s \in \mathcal{S}$,
- the actuator A taking values $a \in \mathcal{A}$,
- the rest of the environment R taking values $r \in \mathcal{R}$,

which allows to model noise. The relationship between S, A and R, i.e., the perception-action loop, can be expressed as a time-unrolled *Causal Bayesian Network*, where A_t depends on S_t, R_t on A_{t-1}, and S_t on R_t. The perception-action loop can be understood as a probabilistic channel, and empowerment is defined as the channel capacity between the agent's actuator A_t and sensor S_{t+1}:

$$\mathrm{E} := C(A_t \rightarrow S_{t+1}) = \max_{p(a_t)} I(S_{t+1}, A_t), \tag{1}$$

where $I(S_{t+1}, A_t)$ is the mutual information between two random variables [19]:

$$I(S_{t+1}, A) := \sum_{s \in \mathcal{S}} p(s) \sum_{a \in \mathcal{A}} p(a \mid s) \log p(a \mid s) - \sum_{s \in \mathcal{S}} p(s) \log p(s). \tag{2}$$

The first term corresponds to the conditional entropy, and the second term to the standard Shannon entropy. The Shannon entropy measures the uncertainty of a random variable. The conditional

entropy measures the uncertainty in S_{t+1} once A is known. Mutual information then measures the average information one can gain about S_{t+1} by observing A [19]. Thus, empowerment can be understood as a measure of how much *influence* an agent has on its environment given its actuator and sensor. If, e.g., all actions result in the same states, then the agent has no empowerment. The same is true if the actions have no perceivable influence on the environment, e.g., when the environment changes randomly. Thus, the effect of the actuators has to be perceivable by the sensor.

From the formal definition of empowerment in Eq. (2) we see that the agent needs a causal model of how its action A_t in the current state of the environment R_t influences the sensor reading S_{t+1}. A model is needed that tells the agent how its actions influence future readings for the actual states of the world. Such a causal model can be obtained efficiently from observations [10].

It is important to note that empowerment represents the *potential* information flow. The agent calculates how it *could* affect the environment, and does not materialize its potential [20]. A way to act upon empowerment could be to greedily select an action resulting in the state with highest empowerment.

Empowerment provides a number of desirable properties [18]:

- *It is* **agent-centric:** Only information accessible to the agent is used, i.e., samples from the perception action loop (sensimotoric data).
- *It features* **locality:** No global knowledge of the world is necessary.
- *It is* **well-defined and computable:** Due to the channel formulation, standard information-theoretic quantities and established methods can be used for its calculation
- *It is* **semantically unbiased:** No external reward system is introduced.

Especially the last point sets empowerment apart from traditional Reinforcement Learning (RL) approaches [16, 17] and mathematical optimization [2, 14, 21]. In RL and optimization, the designer has to define a specific reward signal or objective (e.g., related to Quality of Service or Quality of Experience measures). Defining a specific objective function is often non-trivial. The objective might be multi-dimensional and potentially depends on many aspects (e.g., on routing latency, resilience, etc.). In contrast, empowerment depends only on the agent's embodiment and the environment. This makes empowerment easily applicable to self-driving networks since, no extensive world (or network) model is required, and empowerment can be used to maximize future options. Specific measures like latency or throughput might implicitly be optimized as well [13].

2.2 Application: Reconfigurable Networks

We are concerned with the question of how to leverage empowerment in the context of (self-driving) communication networks. We specify the different components: the environment, actions and sensors, for a specific application in networking: *routing in reconfigurable networks*. We choose this case study because it is rather general (allowing not only to select and adapt routing requests but also edges) and challenging; hence, it shows the potential and limitations of empowerment. Moreover, reconfigurable networks are an emerging and not well-understood paradigm [4, 6, 7].

The Context. We model the network as a capacitated graph $G = (\mathcal{N}, \mathcal{E}, b, c)$, where the function $b : \mathcal{E} \to \mathbb{N}$ gives the capacity of an edge in routing units, and the function $c : \mathcal{E} \to \mathbb{R}$ the cost of using an edge. We also have a set of *possible* routing requests (demand) $\mathcal{D}$. We model a request as a source-destination pair given by the triple (s, t, d), where $s \in \mathcal{N}$ is the source, $t \in \mathcal{N}$ the destination and $d \in \mathbb{N}$ the requested capacity (in routing units). A subset $\mathcal{D}' \subseteq \mathcal{D}$ is eventually revealed, and must then be served by the network.

We consider reconfigurable networks whose edges can be adjusted, e.g., using free-space optics in datacenters [6, 7] or reconfigurable optical switches in Wide-Area Networks [4, 8]. Similar to [6] and [8], we allow the formation of multiple edges between any pair of nodes, assume equal capacity on all edges, and restrict the maximum number of edges incident to a node v to be at most k. Reconfigurable topologies are helpful in the face of changing traffic patterns, since they allow the network to adapt to a new pattern, and thus mitigate e.g., congestion [6] or increase throughput [8].

The Problem. The design of a topology that can serve as many requests with as few reconfigurations as possible. We impose a maximum number of reconfigurations that can be performed due to the effort of reconfigurations. The topology should then be designed such that (1) many requests can immediately be served, and (2) reconfigurable resources can be used to maximum effect (a similar problem is considered in [6]).

The Solution. Empowerment presents itself naturally as a solution to this problem. Here an agent uses empowerment to restructure the network, consequently actions correspond to the configuration of edges, and the sensor relates to routed flows or packets. Intuitively, an agent has high empowerment if the edges it can reconfigure result in many different sets of routed requests.

The Challenge. The actuators and sensors for the agent need to be formulated with care. If, for example, the changes made in the environment are not perceivable with the sensor, then the agent has empowerment 0 (by Eq. (2)) and the approach is bound to fail. If the actuators and sensors are too complex, i.e., have high dimensionality, then calculation of empowerment can be computationally expensive since the perception-action loop needs to be sampled many times, which can be costly. The calculation of empowerment must be fast to allow the agent to react quickly.

The Environment. Includes the graph G as well as the routed requests. Thus, for a given demand $\mathcal{D}$, the environment is defined as $\mathcal{R} := \{(\mathcal{D}_r, \mathcal{E}') \mid \mathcal{D}_r \in 2^{\mathcal{D}}, \mathcal{E}' \in 2^{\mathcal{N}\times\mathcal{N}}\}$, where $\mathcal{D}_r$ is the set of routed requests and $\mathcal{E}'$ the set of realized edges.

The Agent is equipped with a set of actuators and one sensor, which together form its embodiment. We consider one single agent that influences the environment (edges and routed requests) through its actuators and makes observations through its sensors.

The Actuators: To achieve a large empowerment, the agent's actuators must be able to have an impact on the environment. We define five actuators:

- EdgePlacer (EP) establishes an arbitrary edge in the network, taking values $\mathcal{A}_{EP} := \mathcal{N} \times \mathcal{N}$. If an existing edge is chosen, the placement of the edge corresponds to an increase of the capacity on that edge.
- EdgeRemover (ER) removes an arbitrary existing edge. It takes values $\mathcal{A}_{ER} := \{e \mid e \in \mathcal{E}'\}$.

- RequestPlacer (RP) chooses an arbitrary request from $\mathcal{D}$ that is not yet routed, $\mathcal{A}_{RP} := \mathcal{D}/\mathcal{D}_r$, and tries to find a shortest path in the graph. On success, the demanded resources are allocated.
- RequestRemover (RR) removes an arbitrary routed request from the graph, $\mathcal{A}_{RR} := \mathcal{D}_r$.
- Idler corresponds to "*do nothing*"; this actuator does not change the environment.

The Sensors: Similar to the actuators, if empowerment is to be meaningful, the sensors must relate to the environment. We define the following two sensors as functions of an environment state $r \in \mathcal{R}$ and a positive number $l \in \mathbb{R}^+$:

- ExactRequests (EQR): $\text{EQR}(r, l) := \mathcal{D}_r$ if $\mid \mathcal{D}_r \mid > l$ else $\emptyset$ returns the set of currently realized requests, if the number of realized requests is larger than a specific value. Else, the empty set is returned. The size of $\mathcal{S}$ grows exponential with the maximum number of requests that can be routed.
- NumRequests (NQR): $\text{NQR}(r, l) :=\mid \mathcal{D}_r \mid$ if $\mid \mathcal{D}_r \mid > l$ else 0, returns the number of currently realized requests, if that number is larger than l. Else zero is returned. The size of $\mathcal{S}$ grows linear with the maximum number of requests that can be routed.

The intuition behind the sensor threshold l is to help the agent to cope with the combinatorial complexity of the problem. The method to approximate empowerment of a state in the environment, introduced in the next section, is limited and has difficulties when $\mathcal{S}$ is large. Threshold adaptation helps the agent to cope with the large space of sensor readings, as we will show later.

A sensor reading is obtained by sequentially performing shortest path routing for all requests from demand $\mathcal{D}$. To mitigate the effect of request ordering, we take the sensor reading corresponding to the largest amount of routed requests out of ten random permutations or subsets of $\mathcal{D}$.

We examine four embodiments in our evaluation: ExactBuilder (`EB`), SimpleBuilder (`SB`), ExactController (`EC`) and SimpleController (`SC`) with the following actions/sensor:

- `EB` := ({EP, ER, Idler}, EQR)
- `SB` := ({EP, ER, Idler}, NQR)
- `EC` := ({EP, ER, Idler, RP, RR}, EQR)
- `SC` := ({EP, ER, Idler, RP, RR}, NQR)

`Exact` indicates that the embodiment uses the EQR sensor and `Simple` the NQR one. `Builder` indicates that the agent has only control over the graph of the network, i.e., can place and remove edges. In contrast, the `Controller` embodiment has additionally control over the routed requests. `Builder` embodiments can influence their corresponding sensor only indirectly by placing or removing edges and thus enabling or disabling requests. The `Controller` embodiments have direct influence on their sensor due to their ability of adding/removing requests.

3 PRELIMINARY EXPERIMENTS

In order to shed first light on the potential benefits but also limitations of using empowerment in networks, we conducted a case study of a self-driving network which adjusts to the routing requests it has to serve. We consider a simple discrete system where an agent chooses, at each time step, the action maximizing empowerment.

3.1 Algorithms

One of the main objectives considered in literature is to maximize the number of accepted and routed requests in the network [6, 8]. Thus, for comparison, we introduce an Integer-Linear-Program (ILP) that serves as an optimal baseline. Since ILPs are computationally expensive, we introduce a Simulated Annealing (SA) heuristic. Due to space we keep the algorithm descriptions short, but will release our source code to aid reproducibility[1].

Exact Baseline (ILP). The problem we are considering can be modeled as a robust optimization problem. Let $\mathcal{D}$ be the set of all possible demands, and $f : \mathcal{D}, \mathcal{N} \times \mathcal{N} \to \mathcal{D}$ be a function which, given a set of requests and a specific configuration of edges, returns all requests that can be routed. Given a set $\boldsymbol{D}_{RO} \subseteq 2^{\mathcal{D}}$ of requests that should be served, the robust objective is then:

$$\max_{\mathcal{D}' \in \boldsymbol{D}_{RO}} \left(\min_{\mathcal{E} \in 2^{\mathcal{N} \times \mathcal{N}}} \left| \mathcal{D}' / f(\mathcal{D}', \mathcal{E}) \right| \right), \tag{3}$$

subject to additional constraints. We assume $\boldsymbol{D}_{RO} := 2^{\mathcal{D}}$, then optimizing Eq. (3) trivially reduces to:

$$\min_{\mathcal{E} \in 2^{\mathcal{N} \times \mathcal{N}}} \left| \mathcal{D} / f(\mathcal{D}, \mathcal{E}) \right|. \tag{4}$$

Given this fact, our goal is then to compute a network which maximizes the number of routed requests, while minimizing the bandwidth cost as a secondary objective:

$$\min \sum_{h \in \mathcal{D}} \sum_{e \in \mathcal{E}} c(e) \cdot x_h(e) - \gamma \sum_{h \in \mathcal{D}} r(h), \tag{5}$$

where $r(h)$ is a binary variable, which is one if demand h is routed. $x_h(e)$ is a binary variable which is one if demand h is routed along edge e, and γ is a large positive constant, e.g., $\gamma > \sum c(e)$.

For the sake of modeling, we extend the undirected graph G with unit cost and zero capacity ($b(e) = 0$) edges to a full-mesh graph G^M. In G^M every edge is replaced by two anti-parallel arcs with identical cost and capacity values with its corresponding (undirected) edge. The following constraints are required:

$$\forall h \in \mathcal{D}, \forall i \in \mathcal{N}:$$

$$\sum_{j \in \mathcal{N}:(i,j) \in \mathcal{E}} x_h(i,j) - \sum_{j \in \mathcal{N}:(j,i) \in \mathcal{E}} x_h(j,i) = \begin{cases} r(h), \text{if } i = s \\ -r(h), \text{if } i = t \\ 0, \text{else} \end{cases}, \tag{6}$$

$$\forall (i,j) \in \mathcal{E} : \sum_{h \in \mathcal{D}} d_h \left[x_h(i,j) + x_h(j,i) \right] \leq p(i,j), \tag{7}$$

$$\forall (i,j) \in \mathcal{E} : p(i,j) = p(j,i), \tag{8}$$

$$\forall i \in \mathcal{N} : \sum_{j \in V:(i,j) \in \mathcal{E}} \frac{p(i,j) - b(i,j)}{c(i,j)} \leq u, \tag{9}$$

$$\sum_{e \in \mathcal{E}} \left[p(e) - b(e) \right] \leq 2I, \tag{10}$$

$$\sum_{e \in \mathcal{E}} \left| p(e) - b(e) \right| \leq 2B. \tag{11}$$

Eq. (6) formulates the flow conservation for every routed demand. The additionally deployed capacity $p(e)$ is set in Eq. (7)-(8) for every

[1] https://github.com/tum-lkn/empowernets

undirected edge in the original graph G by summing up the flow values on its corresponding directed arcs. The total number of reconfigurable edges per node is bounded in Eq. (9), where $u = kb(e)$ with $e \in \mathcal{E}$ is a multiple of the bandwidth, which is the same for all edges. In Eq. (10)-(11) the capacity increase I and reconfigurations B are bounded (multiplied by two because we have to perform them on both bi-directional arcs), respectively.

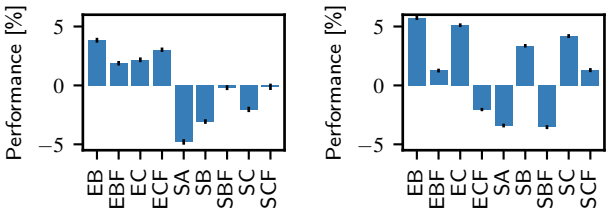
Simulated Annealing (SA). Is a heuristic approximation method motivated by the Metropolis-Hastings algorithm [12]. We use it to place and move edges in an initial topology in such a way that the number of routed flows is maximized. The algorithm works as follows: Starting from an initial solution (any set of edges), it randomly decides to place or remove an edge. The probability of the two events depends on the number of edges that can still be deployed. We will refer to this number as inventory. If the inventory is empty an edge is taken with high probability and vice verca. In case of an edge placement the algorithm filters out all those nodes in the network that already have a maximum degree k. The starting node of the edge is sampled with probability proportional to the number of request sources and destinations located at each node. Only nodes with a degree smaller than the limit are considered. A target node is then drawn uniformly at random from all requests that have the source node of the edge as source. In case of an edge removal any existing edge is removed uniformly at random.

Empowerment Algorithm. We will focus on a discrete and deterministic world model: We assume state transitions are deterministic and sensor readings, as well as actions are discrete. In this case Eq. (2) reduces to the logarithm of perceivable sensor readings given an initial state [20].

We also note that the characteristics of the agent-environment interaction, that is, the effect of placing or removing an edge, might become distinguishable only after several steps. Accordingly, we consider *n-step* empowerment: We consider not a single action A_t but a sequence of variables for the n next time-steps, $(A_t, \ldots, A_{t+n})$, and consider only the sensor reading S_{t+n+1} [19].

With increasing sequence length becomes the computation of exact empowerment values quickly intractable, since the evaluation of $|\mathcal{A}|^n$ action sequences would be required. Therefore, we use *sparse sampling* [20] to approximate n-step empowerment. In sparse sampling m action sequences of length n are sampled at random, and the m sensor readings S_{t+n+1} are used to approximate empowerment. Sparse sampling works under the assumption that the number of all perceivable sensor readings $|\mathcal{S}|$ is small compared to the number of ways to obtain them. This assumption is not met by the EQR sensor, during the evaluation we will show how filter adaptation can mitigate this limitation.

Since empowerment represents how the agent could potentially affect the environment, we need a way to actually choose an action. We consider here a greedy policy that chooses the action resulting in the state with maximum n-step empowerment. To select an action we thus have to perform $m \mid \mathcal{A} \mid$ action sequences once, and can store the result in a table for later reuse. Empowerment at a specific state is computed using the actual world model, that is the agent knows exactly how the world will develop.



(a) $n = 5$ **(b)** $n = 15$

Figure 2: Accepted requests in percent relative to the ILP for sequence length $n = 5$ and $n = 15$. A positive value indicates an improvement, a negative value a degradation compared to the ILP. Error bars shows the 95 % confidence interval.

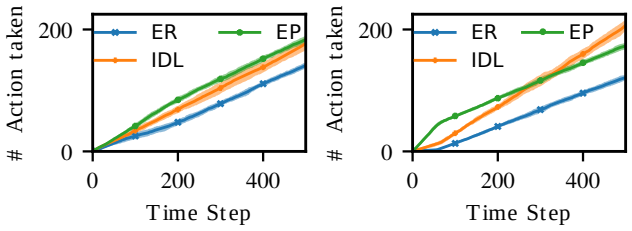
3.2 Evaluation

We investigate how greedy optimization of empowerment relates to solutions from ILP and SA, and the effect of threshold adaptation. We consider an undirected graph with 30 nodes. Each node has at most $k = 3$ reconfigurable edges, resulting in a maximum number of 45 edges. Edges have unit capacity and cost: $\forall e \in \mathcal{E} : b(e) = 1, c(e) = 1$, and we consider as potential requests $\mathcal{D}$ all s-t pairs with unit demand, i.e., $\mathcal{D} := \{(s, t, 1) \mid (s, t) \in \mathcal{N} \times \mathcal{N} \wedge s \neq t\}$, resulting in $|\mathcal{D}| = \frac{1}{2} |\mathcal{N}| (|\mathcal{N}| - 1) = 435$ requests.

All agent embodiments start from an empty network, and maximize n-step empowerment for $t_{max} = \max(500, \hat{t} + 100)$ time steps, where $\hat{t}$ is the time step in which the most recent best empowerment value was obtained. In this way early stopping is avoided, i.e., stopping the agent even though there might be room for improvement. In addition, we vary the sensor threshold l. In one set of experiments we keep $l = 0$ constant, in another set of experiments the agent dynamically adapts the threshold by setting it to the average number of accepted requests observed over the executed action sequences. Threshold adaptation is abbreviated with an F, i.e., `ECF` indicates the usage of the `ExactController`-Embodiment, where the sensor threshold l is dynamically adapted. ILP and SA maximize the number of accepted requests out of $\mathcal{D}$. We give the ILP an infinite action budget for this purpose.

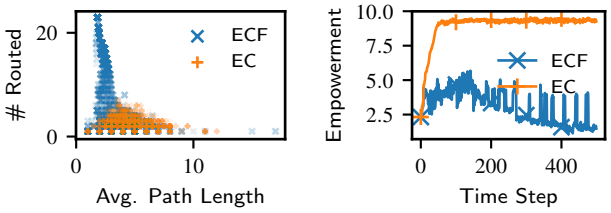
After obtaining topologies from agent embodiments, ILP and SA, we evaluate the "preparedness" of these topologies by drawing 100 demands $\mathcal{D}'_1, \ldots, \mathcal{D}'_{100}$ having 45 requests each without replacement uniformly at random from $\mathcal{D}$. We then use the ILP with action budgets of $B \in \{5, 15\}$ to get the maximum number of routable requests for each $\mathcal{D}'_i$ for each topology. The action budget B and sequence length n coincide, e.g., for $B = 5$ we consider topologies created from agents with $n = 5$.

Comparing Preparedness. Fig. 2 shows the difference in accepted requests as percent relative to the ILP. A simple greedy policy for empowerment maximization can indeed result in topologies, which are able to accept more requests. Fig.2a shows that for $n = 5$ the topology designed by the `EB` is able to accept 4 % more requests on average compared to the ILP, and 5 % more for $n = 15$. Embodiments with the EQR sensor always outperform the ILP. In contrast, Fig. 2a shows that embodiments with the NQR sensor always perform worse for $n = 5$, while for $n = 15$ (Fig. 2b) `SB` and `SC` outperform the ILP. All embodiments always perform at least as good as SA.



(a) EB, $l = 0$ **(b) EB, l adaptive**

Figure 3: Chosen actions and empowerment for the EB with sequence length $n = 10$.



(a) Routing Quality **(b) Empowerment**

Figure 4: Number of routed requests and average path length for EC with and without filtering, with $n = 10$.

This analysis shows that depending on the embodiment empowerment-driven agents are indeed able to structure a network towards the specific purpose of accepting varying future demands.

Chosen Actions. Do agents show a behavior in the choice of actions? Fig. 3 shows how often the EB and EBF chooses an action as cumulative sum over time, averaged across ten executions. The shaded area corresponds to one standard deviation. The results are exemplary and similar for the respective actuators across all embodiments. The agent chose actions with purpose. Fig. 3a and 3b show that the corresponding agent prefers the placement of edges during the first 100 time steps. After that, the curve for the Idler actuator increases, indicating that the agents actively do nothing. This behavior is more pronounced for the EB with threshold adaptation. The curves for EP and ER become parallel, indicating that they are sampled in equal measure due to random tie breaks.

Thus, the agent displays purpose behind the choice of actions. In the beginning edge placement increases empowerment, since requests can be routed, resulting in different sensor readings. The EBF requires an increasing number of routed requests to obtain new readings due to threshold adaptation, and is thus more eager to place edges. After a certain time, empowerment cannot be improved any further by placing edges, and the agents idle or take actions at random due to tie breaks. This behavior is similar across all embodiments. Fig. 4b shows a prototypical development of empowerment for the EC, where empowerment strongly increases during the first 100 time steps and then flattens out.

Sensor Adaptation and Request Selection. To understand how threshold adaptation impacts selection of routed requests and topology design, we investigate the average path length and number of routed requests in the graph for the EC and ECF with sequence length $n = 10$. We chose the average path length because it indicates how well the requests picked by EC and ECF utilize the bandwidth resources. Fig. 4a plots the average path length against the number of routed requests, and shows that threshold adaptation leads to an increase in routed requests, and a decrease in the path length. Thus the ECF learned to place edges and choose requests such that requests are routed with low resource footprint.

Filter adaptation thereby addresses two issues: (1) The indifference of agents wrt. the number of routed requests, and (2) limits of sparse sampling used to approximate n-step empowerment. The empowerment for the EC is 9.88 corresponding to 942 sensor readings, which is close to the possible 1 000 readings of sparse sampling. Increasing the number of action sequences m can in theory counteract this problem. However, due to the indifference of empowerment with respect to the size of routed request sets a very large number of sequences would be needed. An agent maximizing empowerment is not concerned with increasing the number of requests in the network if this does not result in more readings. Due to the agents' indifference, the number of possible readings is doubled with each request that can be routed, i.e, grows exponentially.

Sensor adaptation guides the agent toward a specific region of the state space by shaping the empowerment landscape. Fig. 4b depicts the development of empowerment as observed by the agent for the EC and ECF. Fig. 4b shows that empowerment reaches a plateau close to 10 after 100 time steps for the EC. In contrast, empowerment decreases after an initial surge for the ECF, fluctuates, and always stays well below the curve of the EC.

The fluctuations are caused by threshold adaptation, which reduces the number of sensor readings, resulting in lower estimated n-step empowerment. This reduction helps the agent to overcome the limitations of sparse sampling and actually increases its empowerment further: If $|\mathcal{D}_r| = 10$ then the agent can obtain 2^{10} different readings (the threshold of sparse sampling) within a time horizon of $n = 1$. If $|\mathcal{D}_r| = 20$ then $2^{20} >> 2^5$ readings are possible, an effect that becomes more pronounced with increasing n. Since we are using the average over the size of observed routed requests, the threshold will saturate and not increase endlessly. Threshold adaptation thus enables the usage of sparse-sampling and the benefit of reduced computational effort in a large sensor space.

4 CONCLUSION

We understand our work as a first step in the application of empowerment to self-driving networks and believe that empowerment opens many interesting directions for future research. So far we have focused on a single and simplistic case study only. We plan to extend this case study towards realistic scenarios and additional aspects, utilizing real world demands and networks. For instance, we think that it would be interesting to use empowerment in faulty networks, where e.g., links or middle-boxes might fail, and to investigate the impact empowerment maximization has on traditional Quality of Service and Quality of Experience measures.

ACKNOWLEDGEMENT

This work is part of a project that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program (grant agreement No 647158 - FlexNets).

REFERENCES

[1] Tom Anthony, Daniel Polani, and Chrystopher L. Nehaniv. 2011. Impoverished Empowerment: 'Meaningful' Action Sequence Generation through Bandwidth Limitation. In *Advances in Artificial Life. Darwin Meets von Neumann*, George Kampis, István Karsai, and Eörs Szathmáry (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 294–301.

[2] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Räcke. 2004. Optimal oblivious routing in polynomial time. *J. Comput. System Sci.* 69, 3 (2004), 383–394.

[3] D. Bertsimas, D. Brown, and C. Caramanis. 2011. Theory and Applications of Robust Optimization. *SIAM Rev.* 53, 3 (2011), 464–501.

[4] S. Das, G. Parulkar, and N. McKeown. 2013. Rethinking IP core networks. *IEEE/OSA Journal of Optical Communications and Networking* 5, 12 (Dec 2013), 1431–1442.

[5] Nick Feamster and Jennifer Rexford. 2017. Why (and How) Networks Should Run Themselves. *arXiv preprint arXiv:1710.11583* (2017).

[6] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel Kilper. 2016. ProjecToR: Agile Reconfigurable Data Center Interconnect. In *Proc. ACM SIGCOMM*. ACM, New York, NY, USA, 216–229.

[7] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R Das, Jon P Longtin, Himanshu Shah, and Ashish Tanwer. 2014. Firefly: A reconfigurable wireless data center fabric using free-space optics. In *Proc. ACM SIGCOMM Computer Communication Review (CCR)*, Vol. 44. 319–330.

[8] Xin Jin, Yiran Li, Da Wei, Siming Li, Jie Gao, Lei Xu, Guangzhi Li, Wei Xu, and Jennifer Rexford. 2016. Optimizing bulk transfers with software-defined optical WAN. In *Proc. ACM SIGCOMM*. 87–100.

[9] M. Karl, J. Bayer, and P. van der Smagt. 2015. Efficient Empowerment. *ArXiv e-prints* (Sept. 2015). arXiv:stat.ML/1509.08455

[10] M. Karl, M. Soelch, P. Becker-Ehmck, D. Benbouzid, P. van der Smagt, and J. Bayer. 2017. Unsupervised Real-Time Control through Variational Empowerment. *ArXiv e-prints* (Oct. 2017). arXiv:stat.ML/1710.05101

[11] Wolfgang Kellerer, Arsany Basta, Peter Babarczi, Andreas Blenk, Mu He, Markus Klugel, and Alberto Martinez Alba. 2018. How to Measure Network Flexibility? A Proposal for Evaluating Softwarized Networks. *IEEE Communications Magazine* (2018).

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* 220, 4598 (1983), 671–680.

[13] A. S. Klyubin, D. Polani, and C. L. Nehaniv. 2005. Empowerment: a universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, Vol. 1. 128–135 Vol.1.

[14] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. 2018. Semi-Oblivious Traffic Engineering: The Road Not Taken. In *NSDI 18*. USENIX Association, Renton, WA, 157–170.

[15] A. Leu, D. Ristić-Durrant, S. Slavnić, C. Glackin, C. Salge, D. Polani, A. Badii, A. Khan, and R. Raval. 2013. CORBYS cognitive control architecture for robotic follower. In *Proceedings of the 2013 IEEE/SICE International Symposium on System Integration*. 394–399.

[16] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. 2016. Resource Management with Deep Reinforcement Learning. In *HotNets '16*. ACM, New York, NY, USA, 50–56.

[17] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *SIGCOMM '17*. ACM, New York, NY, USA, 197–210.

[18] Phillipe Capdepuy. 2010. *Informational Principles of Perception-Action Loops and Collective Behaviours*. Phd Dissertation. University of Hertfordshire.

[19] Christoph Salge, Cornelius Glackin, and Daniel Polani. 2013. Empowerment - an Introduction. *CoRR* abs/1310.1863 (2013).

[20] Christoph Salge, Cornelius Glackin, and Daniel Polani. 2014. Changing the Environment Based on Empowerment as Intrinsic Motivation. *CoRR* abs/1406.1767 (2014).

[21] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. 2011. Network architecture for joint failure recovery and traffic engineering. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*. ACM, 97–108.