Decentralized Collective Learning for Self-managed Sharing Economies

EVANGELOS POURNARAS, PETER PILGERSTORFER, and THOMAS ASIKIS,
Professorship of Computational Social Science ETH Zurich, Zurich, Switzerland

The Internet of Things equips citizens with a phenomenal new means for online participation in sharing economies. When agents self-determine options from which they choose, for instance, their resource consumption and production, while these choices have a collective systemwide impact, optimal decision-making turns into a combinatorial optimization problem known as NP-hard. In such challenging computational problems, centrally managed (deep) learning systems often require personal data with implications on privacy and citizens' autonomy. This article envisions an alternative unsupervised and decentralized collective learning approach that preserves privacy, autonomy, and participation of multi-agent systems self-organized into a hierarchical tree structure. Remote interactions orchestrate a highly efficient process for *decentralized collective learning*. This disruptive concept is realized by I-EPOS, the *Iterative Economic Planning and Optimized Selections*, accompanied by a paradigmatic software artifact. Strikingly, I-EPOS outperforms related algorithms that involve non-local brute-force operations or exchange full information. This article contributes new experimental findings about the influence of network topology and planning on learning efficiency as well as findings on techno-socio-economic tradeoffs and global optimality. Experimental evaluation with real-world data from energy and bike sharing pilots demonstrates the grand potential of collective learning to design ethically and socially responsible participatory sharing economies.

CCS Concepts: • **Networks** → **Network structure**; **Network manageability**; **In-network processing**; **Cyber-physical networks**; **Peer-to-peer networks**; *Network management*; • **Computing methodologies** → **Planning for deterministic actions**; **Multi-agent planning**; **Cooperation and coordination**; **Distributed algorithms**; *Multi-agent systems*; • **Computer systems organization** → *Distributed architectures*;

Additional Key Words and Phrases: Computational intelligence, collective intelligence, machine learning, self-management, combinatorial optimization, decentralized system, distributed system, multi-agent system, network, sharing economy, smart grid, smart city

ACM Reference format:
Evangelos Pournaras, Peter Pilgerstorfer, and Thomas Asikis. 2018. Decentralized Collective Learning for Self-managed Sharing Economies. *ACM Trans. Auton. Adapt. Syst.* 13, 2, Article 10 (November 2018), 33 pages.
https://doi.org/10.1145/3277668

Authors' addresses: E. Pournaras, P. Pilgerstorfer, and T. Asikis, Professorship of Computational Social Science ETH Zurich, Zurich, Switzerland, Clausiusstrasse 50, Zurich 8092, Switzerland; emails: {epournaras, peterp, asikist}@ethz.ch.

1556-4665/2018/11-ART10 $15.00
https://doi.org/10.1145/3277668

1 INTRODUCTION

The Internet of Things brings unprecedented opportunities for an alternative paradigm of machine learning that is ubiquitous and decentralized to ultimately preserve citizens' autonomy and privacy. A decentralized and unsupervised collective learning approach designed as a result of self-adaptive and self-organizing actions of autonomous agents promises viable participatory sharing economies emerging in the context of smart grids and smart cities, for instance, energy self-management by prosumers or improvement of urban qualities by citizens via bike sharing initiatives. Such complex techno-socio-economic systems are large in size, online, and involve decision-making processes with combinatorial complexity, i.e., optimization of collective decisions is required to prevent a blackout (Pournaras et al. 2017a; Pournaras and Espejo-Uribe 2017) or to balance the number of bikes in stations (de Chardon et al. 2016). In both cases, a large number of agents coordinate their decision-making to collectively learn how to reduce a power peak or which stations to choose to pick up a bike or return one to keep the utilization of the stations balanced.

When autonomous agents have a set of self-determined options to choose from, e.g., multiple resource consumption and production levels, while the collective outcome of these choices characterizes the overall system performance, the optimization problem is combinatorial and NP-hard in complexity, for instance, the knapsack problem (Puchinger and Raidl 2005). Most earlier work focuses on computational aspects and (meta-)heuristics that efficiently compute solutions by splitting the computational problem into smaller pieces and parallelizing computations. Such approaches include branch and bound based algorithms, for instance, BnB-ADOPT (Yeoh et al. 2008), NCBB (Chechetka and Sycara 2006), and the dynamic programming approach of DPOP (Petcu and Faltings 2005).

In contrast, this article introduces the concept of *decentralized collective learning*, in which participatory agents locally self-determine their options from which they make choices that determine their resource consumption and production. In this context, learning is fully decentralized, and it is the emerging result of coordinated remote interactions that orchestrate collective decision-making over a communication network, self-organized (Pournaras 2013) in a multi-level structure (Deng et al. 2014; Goodfellow et al. 2016), i.e., a tree topology. In this highly challenging scope and problem setting, there is a very limited earlier work, mainly the EPOS (Pournaras 2013; Pournaras et al. 2017b) and COHDA (Hinrichs et al. 2013, 2014) optimization systems, which face significant scalability issues that limit their broader applicability, for instance, EPOS performing expensive non-local brute-force operations, while COHDA requiring a full information exchange between agents.

This article illustrates a generic, unsupervised and highly efficient collective learning algorithm designed to solve fully decentralized combinatorial optimization problems: the Iterative Economic Planning and Optimized Selections (I-EPOS).[1] Agents in I-EPOS autonomously self-determine (i) possible plans that schedule the operations of an application and (ii) their preferences for these plans. The possible plans represent agents' operational flexibility. Agents are structured in self-organized tree topologies (Pournaras 2013) over which they perform collective decision-making in a bottom-up and top-down phase. This process repeats, agents self-adapt their choices and learn new monotonously improving solutions. Information exchange is always at an aggregate level for higher scalability and privacy preservation. The applicability of I-EPOS is studied in two application scenarios of participatory sharing economies: energy management and bike sharing. Synthetic as well as real-world data from state-of-the-art pilot projects are used for the experimental evaluation. Moreover, this article illustrates the implementation of I-EPOS as a open paradigmatic software artifact for promoting further research on decentralized collective learning and optimization as well as the design of new application scenarios.

[1] Available at http://epos-net.org (last accessed: September 2018).

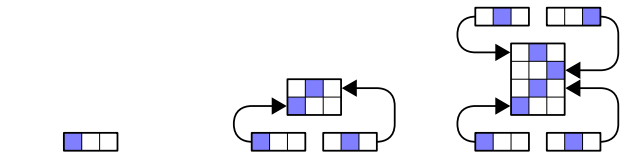


Fig. 1. Plans and responses. An individual box denotes a plan.

Experimental evaluation illustrates striking findings: I-EPOS monotonously and rapidly improves solutions in the order of 10 iterations. A very few number of changes in agents' selections are required to maximize performance. The efficiency of I-EPOS starts from the top 33% of the solution space and converges to solutions below the top 4%. The agents themselves can regulate all performance indicators such as global cost, average local cost, convergence speed, and fairness. The cost-effectiveness in terms of communication and computational cost is superior to related algorithms.

This article is an extension of an earlier work (Pilgerstorfer and Pournaras 2017) with the following additional novel contributions at both conceptual and experimental level:

- Three gradient descent learning schemes evaluated and compared against the main decentralized collective learning scheme of I-EPOS.
- Theoretical insights and analytical results on cost functions and gradient descent learning schemes.
- Measurements of convergence speed under varying number of agents in the network and varying number of children in the tree topology.
- Measurements of global cost and convergence speed under varying number and size of possible plans.
- A study of agent preferences and tradeoffs between global cost, average local cost, convergence speed, and fairness.
- A performance comparison between local and combinational selections in I-EPOS.
- A study of global optimality via comparisons of I-EPOS with an exhaustive brute-force search.

This article is organized as follows: Section 2 formulates the optimization problem and the challenges this article tackles. Section 3 introduces the decentralized collective learning approach of I-EPOS. Section 4 discusses several design aspects of I-EPOS. Section 5 introduces three collective learning schemes based on gradient descent. Section 6 experimentally evaluates I-EPOS, including a performance comparison with three other algorithms and two application scenarios in participatory sharing economies. Section 7 summarizes and discusses all experimental findings and their implications. Section 8 illustrates the implementation of I-EPOS as a paradigmatic software artifact for the broader community. Finally, Section 9 concludes this article and outlines future work.

2 COMBINATORIAL OPTIMIZATION

Table 1 summarizes the mathematical symbols of this article. Assume an *agent* $\mathfrak{a}$ with a finite set of *possible plans* $\mathcal{P}_\mathfrak{a} \subset \mathbb{R}^d$ representing different operational schedules. A *possible plan* $\mathbf{p}_{i,\mathfrak{a}} \in \mathcal{P}_\mathfrak{a}$ is a vector $\mathbf{x}$ of size d with real values representing the allocation of resources, for instance, a time schedule of energy utilization. An agent $\mathfrak{a}$ has to select one and only one possible plan to determine its future operation, the *selected plan*, referred to as $\mathbf{s}_\mathfrak{a}$. Figure 1(a) shows the selected

Table 1. Mathematical Notations Used in This Paper

Notation	Meaning
$\mathcal{A}$	Finite set of all agents in the network
$\mathcal{B}_\mathfrak{a} = \{0, 1\}$	Binary decision for agent $\mathfrak{a}$
$C_\mathfrak{a} \subseteq \mathcal{D}_\mathfrak{a}$	Set of children for agent $\mathfrak{a}$
$\mathcal{D}_\mathfrak{a} \subset \mathcal{A}$	Set of descendants for agent $\mathfrak{a}$
$O(\mathcal{P}_\mathfrak{a}, \mathcal{A}) = \prod_{\mathfrak{a}\in\mathcal{A}} \mathcal{P}_\mathfrak{a}$	All combinations of sets $\mathcal{P}_\mathfrak{a}$ for agents in $\mathcal{A}$
$\mathcal{P}_\mathfrak{a} \subset \mathbb{R}^d$	Possible plans of agent $\mathfrak{a}$
$a = \lvert\mathcal{A}\rvert$	Number of agents
$c = \max_{\mathfrak{a}\in\mathcal{A}} \lvert C_\mathfrak{a}\rvert$	Maximum number of children per agent
$d \in \mathbb{N}^+$	Size of plans
$p = \max_{\mathfrak{a}\in\mathcal{A}} \lvert\mathcal{P}_\mathfrak{a}\rvert$	Maximum number of plans per agent
$t \in \mathbb{N}^+$	Number of iterations
$\mathbf{o} \in O(\mathcal{P}_\mathfrak{a}, \mathcal{A})$	A combination
$\mathbf{o}^\star \in O(\mathcal{P}_\mathfrak{a}, \mathcal{A})$	An optimal combination
$\mathbf{o}_\mathfrak{a} \in \mathbf{o}$	Plan of agent $\mathfrak{a}$ in combination $\mathbf{o} \in O(\mathcal{P}_\mathfrak{a}, \mathcal{A})$
$o_\mathfrak{c} \in o$	Delta value that encodes a branch approval or rejection for child $\mathfrak{c}$ in combination $o \in O(\mathcal{B}_\mathfrak{c}, C_\mathfrak{a})$
$\mathfrak{a} \in \mathcal{A}$	An agent
$\mathfrak{c} \in C_\mathfrak{a}$	A child of agent $\mathfrak{a}$
$\mathfrak{d} \in \mathcal{D}_\mathfrak{a}$	A descendant of agent $\mathfrak{a}$
$\mathfrak{r} \in \mathcal{A}$	Root agent in the tree
$\tau \in \{1, \ldots, t\}$	Number of the current iteration
$\mathbf{p}_{i,\mathfrak{a}} \in \mathcal{P}_\mathfrak{a}$	Possible plan i of agent $\mathfrak{a}$
$\mathbf{s}_\mathfrak{a}^{(\tau)} \in \mathcal{P}_\mathfrak{a}$	Selected plan of agent $\mathfrak{a}$ at iteration τ
$\mathbf{g}^{(\tau)} = \sum_{\mathfrak{a}\in\mathcal{A}} \mathbf{s}_\mathfrak{a}^{(\tau)}$	Global response of the network at iteration τ
$\mathbf{a}_\mathfrak{a}^{(\tau)} = \sum_{\mathfrak{d}\in\mathcal{D}_\mathfrak{a}} \mathbf{s}_\mathfrak{d}^{(\tau)}$	Descendants' aggregated response of agent $\mathfrak{a}$ at iteration τ
$\mathbf{t}_\mathfrak{a}^{(\tau)} = \mathbf{s}_\mathfrak{a}^{(\tau)} + \mathbf{a}_\mathfrak{a}^{(\tau)}$	Aggregated branch response of agent $\mathfrak{a}$ at iteration τ
$\delta_\mathfrak{a}^{(\tau)} \in \{0, 1\}$	Approval or rejection of branch selection for agent $\mathfrak{a}$ at iteration τ
$\tilde{\mathbf{s}}_\mathfrak{a}^{(\tau)}, \tilde{\mathbf{g}}_\mathfrak{a}^{(\tau)}, \tilde{\mathbf{a}}_\mathfrak{a}^{(\tau)}, \tilde{\mathbf{t}}_\mathfrak{a}^{(\tau)}, \tilde{\delta}_\mathfrak{a}^{(\tau)}$	Preliminary $\mathbf{s}_\mathfrak{a}^{(\tau)}, \mathbf{g}^{(\tau)}, \mathbf{a}_\mathfrak{a}^{(\tau)}, \mathbf{t}_\mathfrak{a}^{(\tau)}$, and $\delta_\mathfrak{a}^{(\tau)}$
$\nabla\tilde{\mathbf{x}}^{(\tau)} = \tilde{\mathbf{x}}^{(\tau)} - \mathbf{x}^{(\tau-1)}$	Change of preliminary value from the one of the previous iteration for $\mathbf{x}^{(\tau)} \in \{\mathbf{s}_\mathfrak{a}^{(\tau)}, \mathbf{g}^{(\tau)}, \mathbf{a}_\mathfrak{a}^{(\tau)}, \mathbf{t}_\mathfrak{a}^{(\tau)}\}$
$f_G : \mathbb{R}^d \to \mathbb{R}$	Global cost function
$f_L : \mathbb{R}^d \to \mathbb{R}$	Local cost function
$E_G^{(\tau)} = f_G(\mathbf{g}^{(\tau)})$	Global cost at iteration τ
$E_L^{(\tau)}$	Average local cost at iteration τ
$U^{(\tau)}$	Unfairness at iteration τ
$w : \mathbb{R}^d \to \mathbb{R}$	Preference weight; raises the cost of disliked plans
$\lambda \in \mathbb{R}$	Controls the tradeoff between global and local cost
$\rho_{i,\mathfrak{a}} \in \mathbb{R}$	Dislike of plan i by agent $\mathfrak{a}$

plan as one of three possible plans. Plan generation can be performed with various methodologies that include clustering (Pournaras et al. 2014a), classification (Fröhling 2017), Markov decision processes (Pandey et al. 2016), or model checking of stochastic multiplayer games (Cámara et al. 2015).

Each *agent* $\mathfrak{a} \in \mathcal{A}$ is connected to a *network* consisting of a set of agents $\mathcal{A}$. The selected plans of several agents summed up together by an agent $\mathfrak{a}$ form the *aggregated response* $\mathbf{a}_\mathfrak{a}$ as shown in Figure 1(b). The selected plans of all agents form a *global response* vector $\mathbf{g} = \sum_{\mathfrak{a} \in \mathcal{A}} \mathbf{s}_\mathfrak{a}$ depicted in Figure 1(c). For example, a home appliance controlled by a software agent can have multiple operating modes (possible plans) with different electricity demand over time. Multiple such devices connected to the smart grid result in a total electricity demand $\mathbf{g}$ that is the global response. A global response comes with a *global cost* $E_G = f_G(\mathbf{g})$, where f_G is a *global cost function*. System wide, a global response with low global cost is preferred over one with a high global cost. The agents' objective is to cooperatively select plans that minimize the global cost. Each possible combination $\mathbf{o} \in O(\mathcal{P}_\mathfrak{a}, \mathcal{A}) = \prod_{\mathfrak{a} \in \mathcal{A}} \mathcal{P}_\mathfrak{a}$ consists of one plan $\mathbf{o}_\mathfrak{a} \in \mathcal{P}_\mathfrak{a}$ per agent $\mathfrak{a}$. The optimal combination $\mathbf{o}^\star$ with the minimal global cost is selected. Cost minimization is defined as follows:

$$\mathbf{o}^\star = \underset{\mathbf{o} \in O(\mathcal{P}, \mathcal{A})}{\arg\min} \; f_G\left(\sum_{\mathfrak{a} \in \mathcal{A}, \mathbf{o}_\mathfrak{a} \in \mathbf{o}} \mathbf{o}_\mathfrak{a} \right), \tag{1}$$

$$\mathbf{s}_\mathfrak{a} = \mathbf{o}_\mathfrak{a}^\star \quad \forall \mathfrak{a} \in \mathcal{A}.$$

The number of combinations is $O(p^a)$, where p is the maximum number of plans per agent and a is the number of agents in the network. Optimal solutions are computationally feasible only for a low number of possible plans and low in size networks. The overall problem can be classified as 0-1 multiple-choice combinatorial optimization problem. A related problem is, for example, the 0-1 multiple-choice knapsack problem. In general, these problems are NP-hard, which require approximation algorithms such as I-EPOS, the algorithm introduced in Section 3.

The aforementioned global optimization problem is especially applicable for non-linear global cost functions. For instance, the minimization of variance is a quadratic function (Rockafellar et al. 2000) of the plans size d. Appendix A provides a background on the cost functions studied in this article. The minimization of variance as a non-convex optimization problem (Allen-Zhu and Hazan 2016) is the focus of this article as it can formulate several load-balancing and stability objectives in application domains of sharing economies. For instance, Section 6.5 shows how variance can represent the power peaks in energy demand-response programs or the uneven utilization of bike-sharing stations. All findings of this article can be extended to other (quadratic) cost functions such as the minimization of the root mean square error (Pournaras et al. 2017b) as a matching indicator between a price signal and a resource allocation or matching supply and demand of resources, e.g., energy.

In contrast, linear global cost functions can be solved locally by each agent as proven in Theorem 2.1.

Theorem 2.1. *For the linear global cost function* $f_G(\mathbf{x}) = \mathbf{v}^\top \mathbf{x}$, *where* $\mathbf{v}^\top \mathbf{x}$ *denotes the scalar product of the vectors* $\mathbf{v}$ *and* $\mathbf{x}$, *the selected plan* $\mathbf{s}_\mathfrak{a} = \mathbf{o}_\mathfrak{a}^\star, \forall \mathfrak{a} \in \mathcal{A}$ *computed by the global combinatorial optimization*

$$\mathbf{o}^\star = \underset{\mathbf{o} \in O(\mathcal{P}_\mathfrak{a}, \mathcal{A})}{\arg\min} \; \mathbf{v}^\top \sum_{\mathfrak{a} \in \mathcal{A}, \mathbf{o}_\mathfrak{a} \in \mathbf{o}} \mathbf{o}_\mathfrak{a}, \tag{2}$$

is the selected plan $\mathbf{s}_\mathfrak{a}$ *computed by the local optimization*

$$\mathbf{s}_a = \underset{\mathbf{p}_{i,\mathfrak{a}} \in \mathcal{P}_\mathfrak{a}}{\arg\min} \, \mathbf{v}^\top \mathbf{p}_{i,\mathfrak{a}}, \quad \forall \mathfrak{a} \in \mathcal{A}. \tag{3}$$

Proof. The global optimization problem can be reformulated as the minimization of the sum over a independent cost functions:

$$\mathbf{o}^\star = \underset{\mathbf{o} \in O(\mathcal{P}_\mathfrak{a}, \mathcal{A})}{\arg\min} \; \mathbf{v}^\top \sum_{\mathfrak{a} \in \mathcal{A}, \mathbf{o}_\mathfrak{a} \in \mathbf{o}} \mathbf{o}_\mathfrak{a} = \underset{\mathbf{o} \in O(\mathcal{P}_\mathfrak{a}, \mathcal{A})}{\arg\min} \sum_{\mathfrak{a} \in \mathcal{A}, \mathbf{o}_\mathfrak{a} \in \mathbf{o}} \mathbf{v}^\top \mathbf{o}_\mathfrak{a}. \tag{4}$$

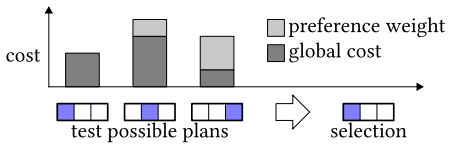


Fig. 2. Agent preferences in plan selection.

Since the summands are independent from each other, the minimization of the sum over all agent selections is equivalent with the minimization of each agent selection:

$$\mathbf{o}^{\star} = \underset{\mathbf{o}\in O(\mathcal{P}_{\mathfrak{a}},\mathcal{A})}{\arg\min} \sum_{\mathfrak{a}\in\mathcal{A},\mathbf{o}_{\mathfrak{a}}\in\mathbf{o}} \mathbf{v}^{\top}\mathbf{o}_{\mathfrak{a}} = \underset{\mathbf{p}_{i,\mathfrak{a}}\in\mathcal{P}_{\mathfrak{a}}}{\arg\min}\, \mathbf{v}^{\top}\mathbf{p}_{i,\mathfrak{a}}, \quad \forall \mathfrak{a} \in \mathcal{A}. \tag{5}$$

Given that the optimization is now performed locally by each agent, the combinatorial search space $O(\mathcal{P}_{\mathfrak{a}}, \mathcal{A})$ is limited to the local possible plans of each agent and therefore it holds that:

$$\mathbf{o}^{\star}_{\mathfrak{a}} = \mathbf{s}_a = \underset{\mathbf{p}_{i,\mathfrak{a}}\in\mathcal{P}_{\mathfrak{a}}}{\arg\min}\, \mathbf{v}^{\top}\mathbf{p}_{i,\mathfrak{a}}, \quad \forall a \in \mathcal{A}. \tag{6}$$

□

Agents' individual preferences for certain plans are controlled via the local parameter λ. A $\lambda = 0$ means no agents' preferences are considered. The larger the λ, the stronger the preferences are. The preference of agent $\mathfrak{a}$ towards a plan is measured by the *local cost function* $f_L(\mathbf{s}_{\mathfrak{a}})$ as depicted in Figure 2. Each agent $\mathfrak{a}$ orders the possible plans $\mathcal{P}_{\mathfrak{a}} = \{\mathbf{p}_{1,\mathfrak{a}}, \mathbf{p}_{2,\mathfrak{a}}, \ldots, \mathbf{p}_{p,\mathfrak{a}}\}$ according to their local cost $f_L(\mathbf{p}_{1,\mathfrak{a}}) \leq f_L(\mathbf{p}_{2,\mathfrak{a}}) \leq \cdots \leq f_L(\mathbf{p}_{p,\mathfrak{a}})$. Plans with lower index are preferred over plans with larger index. The local cost expands the system objective space with the following opportunities:

- Plan selections with a low average local cost E_L, so that plan selections are tuned to meet the agent preferences. The average μ local cost is computed as follows:

$$E_L = \mu\{f_L(\mathbf{s}_{\mathfrak{a}}) \mid \mathfrak{a} \in \mathcal{A}\}. \tag{7}$$

- Plan selections, whose local cost has a low dispersion among agents, so any benefits that agents derive by meeting their preferences are equally distributed among them (Pournaras et al. 2014b). A measure of unfairness U is computed by the standard deviation σ normalized with the mean μ of the local cost for all plan selections:

$$U = \frac{\sigma\{f_L(\mathbf{s}_{\mathfrak{a}}) \mid \mathfrak{a} \in \mathcal{A}\}}{\mu\{f_L(\mathbf{s}_{\mathfrak{a}}) \mid \mathfrak{a} \in \mathcal{A}\}}. \tag{8}$$

  Measurements of fairness can be performed after plan selections by all agents.

3 DECENTRALIZED COLLECTIVE LEARNING

The logical communication network of the agents is assumed (self-)organized in a hierarchical structure: a tree topology. A dynamic and distributed network can be constructed and maintained in a tree topology using AETOS (Pournaras 2013) or the ECHO (Chang 1982) algorithm, for instance. A tree is a acyclic connected graph. It serves the purpose of computing the aggregated and global response in an efficient and accurate way, by preventing double-counting. Moreover, a tree topology provides structured bottom-up and top-down incremental interactions as in the hierarchical structures of neural networks. The *root agent* is denoted as $\mathfrak{r}$. Each agent $\mathfrak{a}$ has a set $C_{\mathfrak{a}}$ of $c = |C_{\mathfrak{a}}|$ *children* and a set of *descendants* $\mathcal{D}_{\mathfrak{a}}$ in a tree branch underneath, with $C_{\mathfrak{a}} \subseteq \mathcal{D}_{\mathfrak{a}}$. An aggregated response $\mathbf{a}_{\mathfrak{a}} = \sum_{\mathfrak{d}\in\mathcal{D}_{\mathfrak{a}}} \mathbf{s}_{\mathfrak{d}}$ of agent $\mathfrak{a}$ corresponds to the descendants' aggregated response in the branch underneath.

The algorithm performs a number of iterations t. Each iteration consists of a *bottom-up* and a *top-down* phase in which the agents change their selected plans to reduce the global cost compared to the previous iteration. Algorithm 1 shows the pseudocode of I-EPOS. The bottom-up and top-down phase for iteration $\tau \in \{1, \ldots, t\}$ are explained below. To simplify the equations, the selected plans at iteration 0 are assumed to be zero: $\mathbf{s}_{\mathfrak{a}}^{(0)} = \mathbf{0}, \quad \forall \mathfrak{a} \in \mathcal{A}$.

ALGORITHM 1: The I-EPOS algorithm.

Input: agent $\mathfrak{a}$, plans $\mathcal{P}_{\mathfrak{a}}$, global cost function f_G, local cost function f_L, parameter λ
Result: selected plan $\mathbf{s}_{\mathfrak{a}}^{(t)}$, aggregated response $\mathbf{a}_{\mathfrak{a}}^{(t)}$, global response $\mathbf{g}^{(t)}$
$\mathbf{s}_{\mathfrak{a}}^{(0)} \leftarrow \mathbf{0}, \quad \mathbf{a}_{\mathfrak{a}}^{(0)} \leftarrow \mathbf{0}, \quad \mathbf{g}^{(0)} \leftarrow \mathbf{0}, \quad \mathbf{t}_{\mathfrak{c}}^{(0)} \leftarrow \mathbf{0} \quad \forall \mathfrak{c} \in C_{\mathfrak{a}}$;
for τ=1 **to** t **do**
  /* BOTTOM-UP PHASE */
  if *agent* $\mathfrak{a}$ *is not a leaf node* **then**
    while *messages from children are missing* **do**
      receive preliminary branch response $\tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}$ from child $\mathfrak{c}$;
    end
    if $\tau = 1$ **then**
      $\tilde{\delta}_{\mathfrak{c}}^{(\tau)} \leftarrow 1, \quad \forall \mathfrak{c} \in C_{\mathfrak{a}}$;
    else
      compute the preliminary deltas $\tilde{\delta}_{\mathfrak{c}}^{(\tau)}$ $\forall \mathfrak{c} \in C_{\mathfrak{a}}$ from Equation (9);
    end
  end
  compute the preliminary aggregated response $\tilde{\mathbf{a}}_{\mathfrak{a}}^{(\tau)}$ and global response $\tilde{\mathbf{g}}_{\mathfrak{a}}^{(\tau)}$ according to Equation (10);
  select preliminary plan $\tilde{\mathbf{s}}_{\mathfrak{a}}^{(\tau)}$ according to Equation (12);
  if *this agent is not the root node* **then**
    send preliminary branch response $\tilde{\mathbf{t}}_{\mathfrak{a}}^{(\tau)} = \tilde{\mathbf{s}}_{\mathfrak{a}}^{(\tau)} + \tilde{\mathbf{a}}_{\mathfrak{a}}^{(\tau)}$ to the parent;
  end
  /* TOP-DOWN PHASE */
  if *this agent is the root node* $\mathfrak{r}$ **then**
    $\mathbf{g}^{(\tau)} \leftarrow \tilde{\mathbf{s}}_{\mathfrak{r}}^{(\tau)} + \tilde{\mathbf{a}}_{\mathfrak{r}}^{(\tau)}$;
    $\delta_{\mathfrak{r}}^{(\tau)} \leftarrow 1$;
  else
    receive global response $\mathbf{g}^{(\tau)}$ and delta value $\delta_{\mathfrak{a}}^{(\tau)}$ from the parent;
  end
  $\delta_{\mathfrak{c}}^{(\tau)} \leftarrow \delta_{\mathfrak{a}}^{(\tau)} \tilde{\delta}_{\mathfrak{c}}^{(\tau)}, \quad \forall \mathfrak{c} \in C_{\mathfrak{a}}$;
  send global response $\mathbf{g}^{(\tau)}$ and delta value $\delta_{\mathfrak{c}}^{(\tau)}$ to each child $\mathfrak{c} \in C_{\mathfrak{a}}$;
  compute selected plan $\mathbf{s}_{\mathfrak{a}}^{(\tau)}$, aggregated response $\mathbf{a}_{\mathfrak{a}}^{(\tau)}$ and branch response $\mathbf{t}_{\mathfrak{c}}^{(\tau)}$ for each child $\mathfrak{c} \in C_{\mathfrak{a}}$ according to Equation (14);
end

3.1 Bottom-up Phase

In this phase, each agent has knowledge about the aggregated response computed as a result of the changes in the selected plans of descendants in the branch underneath the agent. This information propagates from the leaf nodes to the root node. Changes in selected plans of all other agents are

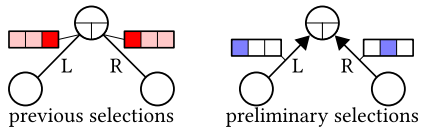


Fig. 3. The input of an agent from its children.

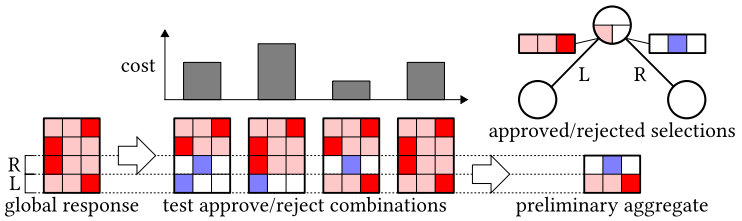


Fig. 4. Approval or rejection of branch selections.

not known. All local decisions made by the agents are *preliminary* at this phase, as the *effective* decisions are made during the top-down phase using knowledge about the parents' decisions. A preliminary plan selection is an actual estimated guess of the optimal one given the incomplete agent knowledge. These guesses are evaluated by the ancestors, who decide which changes of plan selections to approve and which ones to reject. This decision is encoded in the *delta value* $\delta_{\mathfrak{a}}$ of an agent $\mathfrak{a}$, where $\delta_{\mathfrak{a}} = 1$ means the preliminary selection of agent $\mathfrak{a}$ is approved, in contrast to $\delta_{\mathfrak{a}} = 0$ for the rejection of the preliminary selection. In the latter case, the plan selection of the previous iteration remains valid. In the bottom-up phase, each agent $\mathfrak{a}$ receives the preliminary branch response $\tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}$, $\forall \mathfrak{c} \in C_{\mathfrak{a}}$ from its children and computes its own preliminary selection $\tilde{\mathbf{s}}_{\mathfrak{a}}^{(\tau)}$, preliminary aggregated response $\tilde{\mathbf{a}}_{\mathfrak{a}}^{(\tau)}$, preliminary global response $\tilde{\mathbf{g}}_{\mathfrak{a}}^{(\tau)}$ and the preliminary delta values for its children $\tilde{\delta}_{\mathfrak{c}}^{(\tau)}$, $\forall \mathfrak{c} \in C_{\mathfrak{a}}$.

3.1.1 Aggregation. The aggregation of the bottom-up phase aims at summing up the selected plans from the descendants of each agent in the branch underneath that result in the maximal improvement of the global response in comparison to the one of the previous iteration. An agent's knowledge about its children is illustrated in Figure 3.

For each child $\mathfrak{c} \in C_{\mathfrak{a}}$, the agent $\mathfrak{a}$ knows the aggregated response of the child's branch at the previous iteration $\mathbf{t}_{\mathfrak{c}}^{(\tau-1)}$ and receives the preliminary aggregated response of the branch $\tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}$ for the current iteration. The changes from the aggregated response of the previous iteration to the preliminary aggregated response of the current iteration in the branch of child $\mathfrak{c}$ are denoted as $\nabla\tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)} = \tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)} - \mathbf{t}_{\mathfrak{c}}^{(\tau-1)}$. I-EPOS determines which preliminary aggregated response to approve and which to reject by combining knowledge of the branches as shown in Figure 4.

An agent can only approve or reject changes on the aggregated response of a whole branch and not individual plan selections as the latter ones are not known to agents. This makes the algorithm highly decentralized and privacy-preserving in contrast to related work (Hinrichs et al. 2013, 2014) that relies on systemwide exchange of all selected plans (instead of aggregated ones). At the first iteration when there is no history, all changes are approved. In all next iterations, each agent evaluates all possible combinations of branch approvals and rejections $O(\mathcal{B}_{\mathfrak{c}}, C_{\mathfrak{a}})$ and selects a combination that maximally improves the global response. The delta value, that encodes a branch approval or a branch rejection, for child $\mathfrak{c}$ in combination $o \in O(\mathcal{B}_{\mathfrak{c}}, C_{\mathfrak{a}})$ is referred to as $o_{\mathfrak{c}}$. Since an approval is encoded with $o_{\mathfrak{c}} = 1$ and rejection with $o_{\mathfrak{c}} = 0$, the effect of a combination o on the global response is evaluated as follows: $\mathbf{g}^{(\tau-1)} + \sum_{\mathfrak{c} \in C_{\mathfrak{a}}} o_{\mathfrak{c}} \nabla\tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}$. A combination $o^{\star}$ that results in the

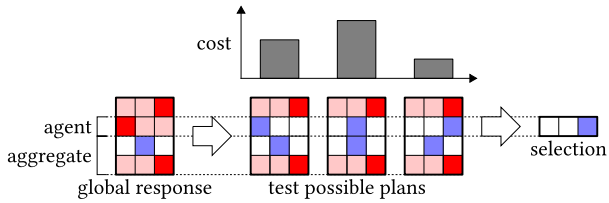


Fig. 5. Preliminary plan selection.

lowest global cost is selected[2]. The delta values of $o^\star$ are chosen as the preliminary deltas $\tilde{\delta}_{\mathfrak{c}}^{(\tau)}$ of the respective child $\mathfrak{c}$. The delta value selection $\tilde{\delta}_{\mathfrak{c}}^{(\tau)}$ can be formalized as follows:

$$o^\star = \underset{o \in O(\mathcal{B}_{\mathfrak{c}}, C_{\mathfrak{a}})}{\arg\min} \; f_G\left(\mathbf{g}^{(\tau-1)} + \sum_{\mathfrak{c} \in C_{\mathfrak{a}},\, o_{\mathfrak{c}} \in o} o_{\mathfrak{c}} \nabla \tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}\right), \tag{9}$$

$$\tilde{\delta}_{\mathfrak{c}}^{(\tau)} = o_{\mathfrak{c}}^\star, \quad \forall \mathfrak{c} \in C_{\mathfrak{a}}.$$

The changes approved by the preliminary deltas are applied to the aggregated response and global response. This results in the preliminary aggregated response and the preliminary global response as shown in the following equations:

$$\tilde{\mathbf{a}}_{\mathfrak{a}}^{(\tau)} = \mathbf{a}_{\mathfrak{a}}^{(\tau-1)} + \sum_{\mathfrak{c} \in C_{\mathfrak{a}}} \tilde{\delta}_{\mathfrak{c}}^{(\tau)} \nabla \tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}, \tag{10}$$

$$\tilde{\mathbf{g}}_{\mathfrak{a}}^{(\tau)} = \mathbf{g}^{(\tau-1)} + \sum_{\mathfrak{c} \in C_{\mathfrak{a}}} \tilde{\delta}_{\mathfrak{c}}^{(\tau)} \nabla \tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}. \tag{11}$$

Note that in the bottom-up phase the children are not aware about the approval or rejection of their preliminary plan selections. This information is sent in the top-down phase.

3.1.2 Plan Selection. Figure 5 illustrates the preliminary plan selection.

Each agent $\mathfrak{a}$ selects a plan that maximally improves the global response. The preliminary global response is used as it includes the preliminary approved changes made by the descendants. The plans are selected[3] as follows:

$$\tilde{\mathbf{s}}_{\mathfrak{a}}^{(\tau)} = \underset{\mathbf{p}_{i,\mathfrak{a}} \in \mathcal{P}_{\mathfrak{a}}}{\arg\min} (1-\lambda) \cdot f_G\left(\tilde{\mathbf{g}}_{\mathfrak{a}}^{(\tau)} + \nabla \mathbf{p}_{i,\mathfrak{a}}\right) + w(\mathbf{p}_{i,\mathfrak{a}}), \tag{12}$$

$$\text{where } \nabla \mathbf{p}_{i,\mathfrak{a}} = \mathbf{p}_{i,\mathfrak{a}} - \mathbf{s}_{\mathfrak{a}}^{(\tau-1)},$$

$$w(\mathbf{p}_{i,\mathfrak{a}}) = \lambda \cdot \rho_{i,\mathfrak{a}} \cdot \sigma\left\{f_G\left(\tilde{\mathbf{g}}_{\mathfrak{a}}^{(\tau)} + \nabla \mathbf{p}_{i,\mathfrak{a}}\right) \mid \mathbf{p}_{i,\mathfrak{a}} \in \mathcal{P}_{\mathfrak{a}}\right\},$$

$$\rho_{i,\mathfrak{a}} = \frac{i}{p},$$

where the preference weight $w(\mathbf{p}_{i,\mathfrak{a}})$ is normalized using the standard deviation of the different global costs for the respective possible plans. The $\rho_{i,\mathfrak{a}}$ expresses the dislike[4] of an agent $\mathfrak{a}$ towards a plan i, given that $f_L(\mathbf{p}_{1,\mathfrak{a}}) \leq f_L(\mathbf{p}_{2,\mathfrak{a}}) \leq \cdots \leq f_L(\mathbf{p}_{p,\mathfrak{a}})$. The $(1-\lambda)$ allows *bounded* input values of λ in the range $[0, 1]$, while omitting $(1-\lambda)$ leaves λ *unbounded* in the range $[0, \infty)$. Both schemes

[2]If multiple combinations are optimal, then $o^\star$ is chosen uniformly at random out of all optimal combinations.
[3]If multiple plans are optimal, then the selected plan is chosen uniformly at random out of all optimal plans.
[4]The $w(\mathbf{p}_{i,\mathfrak{a}})$ in Equation (12) has the limitation of not capturing the actual value of local costs that possible plans have. In other words, the magnitude of the preferences over the possible plans is not captured. Instead $\rho_{i,\mathfrak{a}}$ encodes the index of the plans after sorted according to their local cost as $f_L(\mathbf{p}_{1,\mathfrak{a}}) \leq f_L(\mathbf{p}_{2,\mathfrak{a}}) \leq \cdots \leq f_L(\mathbf{p}_{p,\mathfrak{a}})$.

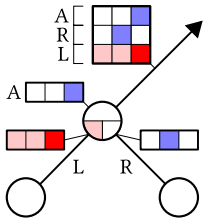


Fig. 6. Agent output during the bottom-up phase.

are evaluated in Section 6.3. Equation (12) is referred to as the *self-adaptive* plan selection scheme. Other schemes in the form of gradient descent are illustrated in Section 5.

3.1.3 Parent Informing. Every non-root agent informs its parent about the selections of its respective branch with the preliminary aggregated response $\tilde{\mathbf{t}}_{\mathfrak{a}}^{(\tau)} = \tilde{\mathbf{s}}_{\mathfrak{a}}^{(\tau)} + \tilde{\mathbf{a}}_{\mathfrak{a}}^{(\tau)}$. This procedure is shown in Figure 6.

3.2 Top-down Phase

In this phase, all agents approve/reject the preliminary selections and they update a consistent for all agents aggregated and global response. At the end of this phase, each agent has a selected plan, the descendants' aggregated response in the branch as well as the global response for the current iteration.

The root agent computes the global response based on its preliminary plans and responses that propagates downwards to all other agents in the network. For the root agent, the preliminary selected plan as well as the preliminary aggregated response correspond to the effective selections: $\mathbf{s}_{\mathfrak{r}}^{(\tau)} = \tilde{\mathbf{s}}_{\mathfrak{r}}^{(\tau)}$ and $\mathbf{a}_{\mathfrak{r}}^{(\tau)} = \tilde{\mathbf{a}}_{\mathfrak{r}}^{(\tau)}$.

The preliminary deltas $\tilde{\delta}_{\mathfrak{c}}^{(\tau)}$ computed are used at this stage to determine the effective delta values $\delta_{\mathfrak{c}}^{(\tau)}$. The selection of the root agent is always approved, hence $\delta_{\mathfrak{r}}^{(\tau)} = 1$. The delta value for the other agents is determined by their respective parent agent $\mathfrak{a}$. The changes of a child $\mathfrak{c}$ are approved if two conditions hold: (i) The ancestors of the parent approve the changes with $\delta_{\mathfrak{a}}^{(\tau)} = 1$. (ii) The parent agent itself approves the changes of the child and its branch with $\tilde{\delta}_{\mathfrak{c}}^{(\tau)} = 1$. Therefore, the parent agent $\mathfrak{a}$ computes the deltas for its children as follows:

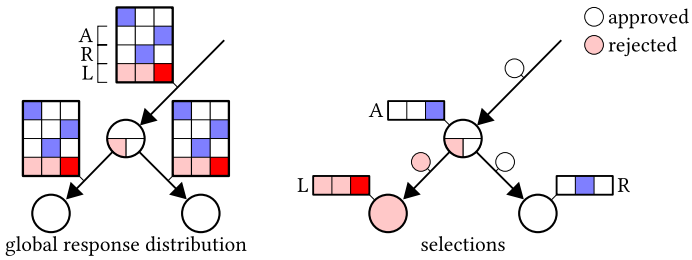
$$\delta_{\mathfrak{c}}^{(\tau)} = \delta_{\mathfrak{a}}^{(\tau)} \tilde{\delta}_{\mathfrak{c}}^{(\tau)}, \quad \forall \mathfrak{c} \in C_{\mathfrak{a}}. \tag{13}$$

A plan selection by an agent is approved in the top-down phase if and only if all its ancestors approve the preliminary plan selection. If one of the ancestors rejects the changes in the responses, then the plan selection of the previous iteration remains as the plan selection of the current iteration. The two scenarios of an approval and a rejection are depicted in Figure 7. Based on the approved changes, the effective plan selections and the aggregated responses are computed as follows:

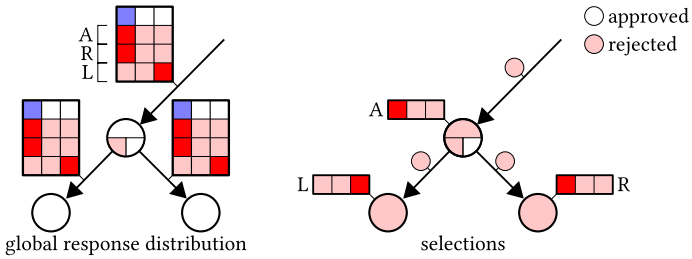
$$\mathbf{s}_{\mathfrak{a}}^{(\tau)} = \mathbf{s}_{\mathfrak{a}}^{(\tau-1)} + \delta_{\mathfrak{a}}^{(\tau)} \nabla \tilde{\mathbf{s}}_{\mathfrak{a}}^{(\tau)}, \tag{14}$$

$$\mathbf{a}_{\mathfrak{a}}^{(\tau)} = \mathbf{a}_{\mathfrak{a}}^{(\tau-1)} + \delta_{\mathfrak{a}}^{(\tau)} \nabla \tilde{\mathbf{a}}_{\mathfrak{a}}^{(\tau)}, \tag{15}$$

$$\mathbf{t}_{\mathfrak{c}}^{(\tau)} = \mathbf{t}_{\mathfrak{c}}^{(\tau-1)} + \delta_{\mathfrak{c}}^{(\tau)} \nabla \tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}, \quad \forall \mathfrak{c} \in C_{\mathfrak{a}}. \tag{16}$$



(a) Ancestors send an approval signal.



(b) Ancestors send a rejection signal.

Fig. 7. Approval and rejection during the top-down phase.

4 DESIGN ASPECTS

Some other design aspects of I-EPOS are discussed below: (i) autonomy, self-determination and participation, (ii) decentralization and privacy preservation, (iii) learning principle and monotonous improvement, and (iv) termination.

4.1 Autonomy, Self-determination, and Participation

Possible plans are self-determined by the agents as they are locally generated without I-EPOS been involved in this process. The learning algorithm does not impose (i) the plans, (ii) their number, or even (iii) the overall participation into the plan generation process at first place. I-EPOS can even operate with agents that have a single possible plan, in other words, inflexible agents that acquire full control of their operation. Even in this extreme case, the decentralized learning process of I-EPOS is designed to be self-adaptive by compensating with the collective decisions of the other participating agents.

In contrast to the vast majority of optimization/learning systems, the design approach of I-EPOS promotes autonomy and self-determination. Free-riding issues and fairness in participation can be addressed in different application scenarios with incentive systems (Li et al. 2013) and reward mechanisms (Scekic et al. 2013).

4.2 Decentralization and Privacy Preservation

Most centralized machine and deep learning techniques operate with all systemwide data available for processing. These data though can be personal and sensitive. Profiling or discriminatory actions (Hajian et al. 2016) can be performed when both personal data and the deep hierarchical structure are managed by a single centralized authority. The majority of deep learning structures have no mapping or association with the data locality and agents' physical distribution.

Unavoidably, this design approach entails risks and ethical implications for the future of artificial intelligence and digital democracy (Helbing et al. 2017; Helbing and Pournaras 2015).

In contrast, this article envisions a digital interconnected society as a fully decentralized and self-organizing neural network. Collective learning processes such as the ones of I-EPOS can be a result of citizens' active participation and wisdom of the crowd. Techniques such as differential privacy (Alaggan et al. 2011) and homomorphic encryption (Parmar et al. 2014) can further enhance privacy-preservation when agents exchange aggregated plans.

4.3 Learning Principle and Monotonous Improvement

In contrast to the earlier self-optimizing approach of EPOS (Pournaras 2013; Pournaras et al. 2017b), I-EPOS extends to a self-adaptive learning process: Plan selections are a collective result of the bottom-up and top-down learning phase at each iteration by using the aggregated responses as well as by using the previous global responses across different iterations.

I-EPOS overcomes a challenging artifact of distributed optimization: the composition of two optimal solutions does not guarantee a combined overall optimal solution, i.e., combining optimization solutions from different tree branches. I-EPOS ensures that independently determined improvements of the global response by different tree branches do not cancel out when aggregated.

To certain extent, the learning concept draws parallels from backpropagation (Bryson and Ho 1969) in neural networks: In the forward pass (bottom-up phase) agents predict their new selections. In the backward pass (top-down phase) the error is backpropagated in the form of delta values. An error of 1 means the agent changes its selection, whereas, an error of 0 means no change is required.

The global cost in one iteration cannot be larger than the global cost of the previous iteration, i.e., the global cost decreases monotonously. Consider the selection process at the root agent. The option to reject preliminary changes ensures that I-EPOS either reduces the global cost or at least maintains the same global cost if the root agent selects the responses of the previous iteration.

4.4 Termination

I-EPOS terminates after a fixed number of iterations to empirically control the communication and computational cost. However, other criteria may be chosen.

- *Termination if no agent performs a plan change.* This solution can be implemented with a boolean flag that agents pass in the top-down phase. This termination approach can cut out some further improvements: agents may choose a different possible plan with equal cost[5] that may lead to new solutions with further improvements in the next iterations. For the same reason, termination is not guaranteed as selections may change without improvement in global cost.
- *Termination once the global cost equals the one of the previous iteration.* This is similar to the previous approach; however, termination is guaranteed given that the global cost decreases monotonously.
- *Termination once the global cost is lower than a threshold.* The two aforementioned termination approaches may result in a large number of executed iterations from which only a few of them result in a significant performance improvement. In this case, a threshold can significantly decrease the communication and computational cost. However, if the threshold is

[5]Recall that more than one optimal possible plans are selected uniformly at random.

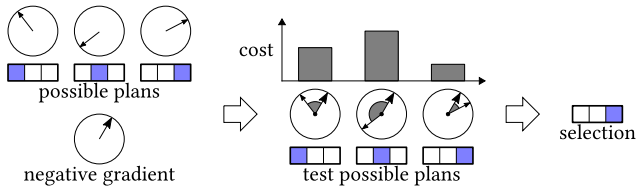


Fig. 8. Gradient descent plan selection.

not chosen effectively or does not match the empirical data in use, it may lead to significant overhead as well.

5 GRADIENT DESCENT COLLECTIVE LEARNING

This section introduces the applicability of gradient descent for the plan selection in I-EPOS. The gradient of the global cost function $\nabla f_G(\mathbf{g}^{(\tau-1)})$ is a vector pointing to the direction with the maximum increase in the global cost. I-EPOS uses the gradient descent to move the global response vector into the negative gradient direction to reduce the global cost. A standard gradient descent update at iteration τ is performed as follows:

$$\mathbf{g}^{(\tau)} \leftarrow \mathbf{g}^{(\tau-1)} - \nabla f_G\left(\mathbf{g}^{(\tau-1)}\right). \tag{17}$$

In the context of gradient descent, I-EPOS solves the non-convex (Allen-Zhu and Hazan 2016) combinatorial optimization problem as follows: Agents select plans that match the negative gradient direction as shown in Figure 8. The collective decisions of I-EPOS make sure that the gradient steps are not too large by determining the subset of agents that change their selected plan in the top-down phase.

Three gradient descent schemes are introduced and evaluated in this article:

(1) **Global gradient:** All agents move the global response to the same negative gradient direction.
(2) **Local gradient:** Each agent performs a different plan selection with the same gradient step.
(3) **Adaptive gradient:** Each agent uses the preliminary global response of Equation (10) with the new selections of all descendants in the tree.

For the global gradient, the preliminary plan selection of Equation (12) is transformed as follows:

$$\tilde{\mathbf{s}}_{\mathfrak{a}}^{(\tau)} = \underset{\mathbf{p}_{i,\mathfrak{a}} \in \mathcal{P}_{\mathfrak{a}}}{\arg\min} \nabla f_G(\mathbf{g}^{(\tau-1)})^{\top} \mathbf{p}_{i,\mathfrak{a}}. \tag{18}$$

For simplicity in the illustration, the term $w(\mathbf{p}_{i,\mathfrak{a}})$ is omitted. The local gradient is computed as follows:

$$\tilde{\mathbf{s}}_{\mathfrak{a}}^{(\tau)} = \underset{\mathbf{p}_{i,\mathfrak{a}} \in \mathcal{P}_{\mathfrak{a}}}{\arg\min} \nabla f_G\left(\frac{a}{a-1}\left(\mathbf{g}^{(\tau-1)} - \mathbf{s}_{\mathfrak{a}}^{(\tau-1)}\right)\right)^{\top} \mathbf{p}_{i,\mathfrak{a}}. \tag{19}$$

Note that the factor $\frac{a}{a-1}$ is required so that the average gradient of the agents is equivalent to the global gradient as proved in Theorem 5.1.

Theorem 5.1. *For quadratic cost functions it holds that the average local gradient is equal to the global gradient:*

$$\frac{1}{a}\sum_{\mathfrak{a}\in\mathcal{A}} \nabla f_G\left(\frac{a}{a-1}\left(\mathbf{g}^{(\tau)} - \mathbf{s}_{\mathfrak{a}}^{(\tau)}\right)\right) = \nabla f_G\left(\mathbf{g}^{(\tau)}\right). \tag{20}$$

Proof. Let a quadratic cost function defined as follows:

$$f_G(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{x}^\top \mathbf{L} + C, \tag{21}$$

with the following gradient:

$$\nabla f_G(\mathbf{x}) = \tilde{\mathbf{Q}}\mathbf{x} + L, \tag{22}$$

where $\tilde{\mathbf{Q}} = (\mathbf{Q} + \mathbf{Q}^\top)$ and $\mathbf{Q}$, $\mathbf{L}$ and C are the quadratic factor, linear factor and constant scalar respectively of a quadratic function as discussed in Section A. The average local gradient that is the first term of Equation (20) can be written in the gradient quadratic form of Equation (22) as follows:

$$\frac{1}{a}\sum_{\mathfrak{a}\in\mathcal{A}}\left(\frac{a}{a-1}\tilde{\mathbf{Q}}\left(\mathbf{g}^{(\tau)} - \mathbf{s}_\mathfrak{a}^{(\tau)}\right) + \mathbf{L}\right). \tag{23}$$

By expanding Equation (23), the following hold:

$$\begin{aligned}
\frac{1}{a}\sum_{\mathfrak{a}\in\mathcal{A}}\nabla f_G\left(\frac{a}{a-1}\left(\mathbf{g}^{(\tau)} - \mathbf{s}_\mathfrak{a}^{(\tau)}\right)\right) &= \frac{1}{a}\sum_{\mathfrak{a}\in\mathcal{A}}\left(\frac{a}{a-1}\tilde{\mathbf{Q}}\left(\mathbf{g}^{(\tau)} - \mathbf{s}_\mathfrak{a}^{(\tau)}\right) + \mathbf{L}\right) \\
&= \sum_{\mathfrak{a}\in\mathcal{A}}\frac{1}{a}\frac{a}{a-1}\tilde{\mathbf{Q}}\left(\mathbf{g}^{(\tau)} - \mathbf{s}_\mathfrak{a}^{(\tau)}\right) + a\frac{1}{a}\mathbf{L} \\
&= \tilde{\mathbf{Q}}\frac{1}{a-1}\sum_{\mathfrak{a}\in\mathcal{A}}\left(\mathbf{g}^{(\tau)} - \mathbf{s}_\mathfrak{a}^{(\tau)}\right) + \mathbf{L} \\
&= \tilde{\mathbf{Q}}\frac{1}{a-1}\left(\sum_{\mathfrak{a}\in\mathcal{A}}\mathbf{g}^{(\tau)} - \sum_{\mathfrak{a}\in\mathcal{A}}\mathbf{s}_\mathfrak{a}^{(\tau)}\right) + \mathbf{L} \\
&= \tilde{\mathbf{Q}}\frac{1}{a-1}\left(a\mathbf{g}^{(\tau)} - \mathbf{g}^{(\tau)}\right) + \mathbf{L} \\
&= \tilde{\mathbf{Q}}\frac{1}{a-1}(a-1)\mathbf{g}^{(\tau)} + \mathbf{L} \\
&= \tilde{\mathbf{Q}}\mathbf{g}^{(\tau)} + \mathbf{L}.
\end{aligned} \tag{24}$$

which is the definition of the gradient quadratic function of Equation (22):

$$\tilde{\mathbf{Q}}\mathbf{g}^{(\tau)} + \mathbf{L} = \nabla f_G\left(\mathbf{g}^{(\tau)}\right). \tag{25}$$

Therefore it is proven that:

$$\frac{1}{a}\sum_{\mathfrak{a}\in\mathcal{A}}\nabla f_G\left(\frac{a}{a-1}\left(\mathbf{g}^{(\tau)} - \mathbf{s}_\mathfrak{a}^{(\tau)}\right)\right) = \nabla f_G\left(\mathbf{g}^{(\tau)}\right), \tag{26}$$

□

Finally the plan selections in the adaptive gradient are performed as follows:

$$\tilde{\mathbf{s}}_\mathfrak{a}^{(\tau)} = \underset{\mathbf{p}_{i,\mathfrak{a}}\in\mathcal{P}_\mathfrak{a}}{\arg\min}\,\nabla f_G\left(\frac{a}{a-1}\left(\tilde{\mathbf{g}}_\mathfrak{a}^{(\tau-1)} - \mathbf{s}_\mathfrak{a}^{(\tau-1)}\right)\right)^\top \mathbf{p}_{i,\mathfrak{a}}. \tag{27}$$

All gradient descent schemes are actually applied in the second iteration, with the first iteration performing initialization. By using the gradient descent, it is not required to compute the global cost for each plan at each iteration. On computation of the gradient, the selected plan can be computed with a linear cost function that has lower computation cost. Moreover, the gradient of possible plans that represents the utilization of resources by agents can be used as price information and therefore can encode monetary incentives (Kim et al. 2011; Traber and Kemfert 2011).

Table 2. Default Experimental Setup

Experimental Parameter	Default Choice
Network topology	Height-balanced binary tree
Number of agents	1000
Dataset	Synthetic Gaussian
Number of possible plans per agent	16
Size of possible plans	100
Parameter λ (unbounded)	0
Global cost	Variance
Number of repetitions	50
Plan selections scheme	Self-adaptive

6 EXPERIMENTAL EVALUATION

Experimental evaluation is performed in a network of 1,000 agents randomly organized in height-balanced binary tree, i.e., a binary tree of minimal height. Each experiment is repeated 50 times with different seeds in the random number generators. Evaluation is performed with both (i) *synthetic* and (ii) *real-world* data from two application domains as illustrated in Section 6.5. The synthetic data concern 16 possible plans of size 100 generated from a standard normal distribution at every experiment repetition. Unless stated otherwise, the synthetic data are used by default as well as agents adopt an unbounded $\lambda = 0$ that deactivates agent preferences over the possible plans. The variance reduction is used as a global cost function that reduces oscillations in the global response. The self-adaptive learning scheme of Section 3 is used for the plan selections by default, which is also compared with the gradient descend schemes of Section 5. The default experimental parameters are outlined in Table 2.

The performance evaluation covers the following six aspects:

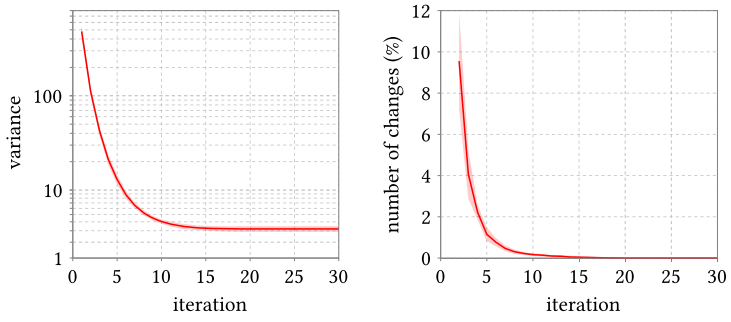
(1) **Convergence and scalability:** Evaluation of the level and speed of global cost reduction under varying system size and tree topology.
(2) **Number and size of plans:** Evaluation of the global cost reduction under varying number and size of possible plans.
(3) **Agent preferences:** Evaluation of the global cost reduction, average local cost and fairness by varying an unbounded and bounded λ parameter.
(4) **Performance comparison:** Comparison of the plan selection schemes as well as comparisons of the computational and communication cost between I-EPOS and three other state of the art methods, namely EPOS, COHDA, and Greedy optimization.
(5) **Application scenarios on sharing economies:** Evaluation of the I-EPOS applicability in energy management and bike sharing.
(6) **Global optimality:** Evaluation of the optimality in the global solutions between I-EPOS and brute-force search.

The rest of this section illustrates each of these performance aspects.

6.1 Scalability and Convergence Speed

Figure 9(a) shows how the global cost, i.e., the variance, changes over the course of 30 iterations. The low in size area around the line is plus/minus the standard deviation of the measured variance over all experiment repetitions.

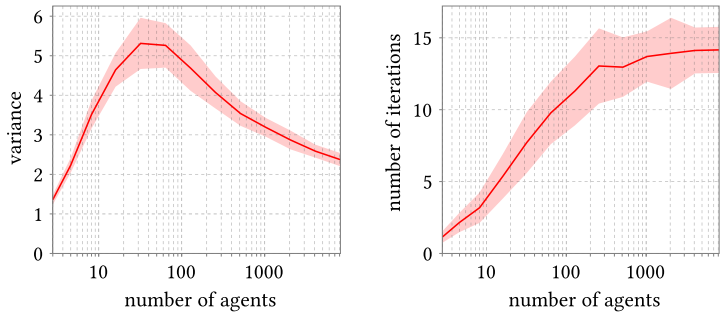
On average, the variance is reduced from 480 ± 70 in the initial iteration to 3.2 ± 0.30 after 30 iterations. As a scale of comparison: randomly choosing standard normally distributed plans



(a) Global cost.

(b) Number of changes in selected plans compared to the previous iteration.

Fig. 9. Variance reduction and changes of plan selections.



(a) Global cost

(b) Convergence speed

Fig. 10. Performance evaluation under increasing number of agents a.

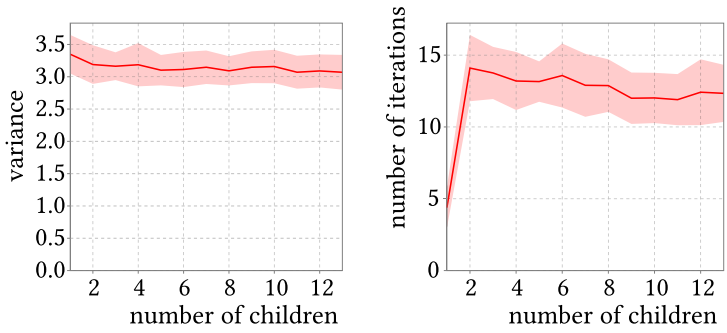
results in an approximate variance of 1000 ± 140 on average.[6] The second termination criterion of Section 4.4 completes the algorithm in 14 ± 2.3 iterations.

Figure 10(a) evaluates the global cost, i.e., variance, and Figure 10(b) the convergence speed, i.e., number of iterations until termination, under varying system size, i.e., incremental increase in the number of agents as $\{2^1, 2^2, \ldots, 2^{13}\}$.

Figure 10(a) shows that a low number of agents results in a low number of combinations and therefore the variance increases as more agents (and therefore more plans) are introduced. After a critical point, the combinations explode and therefore variance starts decreasing as the number of agents increases and a larger solutions space is explored. The convergence speed in Figure 10(b) increases from 2 to 13 iterations as the number of agents scales to $2^{10} = 1{,}000$, and approaches the 15 iterations for larger scales. Note also that the standard deviation increases for higher system scales due to the larger solution spaces.

The computational performance of I-EPOS relies on the topological structure of the tree networks and specifically on the number of children per agent. During the top-down phase, all combinations are evaluated via an exhaustive search with computational complexity of $O(2^c)$. Figure 11 evaluates the effectiveness in reducing variance as well as convergence speed. Given a fixed

[6] The sum of 1,000 standard normally distributed plans is normally distributed with zero-mean and a variance of $\sigma^2 = 1{,}000$. Based on the fact that the empirical variance $\hat{\sigma}^2$ follows a chi-squared distribution $\frac{d-1}{\sigma^2}\hat{\sigma}^2 \sim \mathcal{X}^2_{d-1}$, the empirical variance is expected to be 1,000 with a standard deviation of about 142.



(a) Global cost (b) Convergence speed

Fig. 11. Performance evaluation under increasing number of children c.

network size of 1,000 agents, measurements are made with $c \in \{1, 2, \ldots, 13\}$, with $c = 1$ corresponding to a list of agents and therefore a maximal tree depth.

Figure 11(a) shows that for $c \geq 2$ performance improves at a very low extent while the convergence speed in Figure 11(b) increases for 2 iterations approximately. Therefore, this observation shows that there is no significant gain in variance reduction or convergence speed by increasing the computational complexity via a larger number of children in the tree topology. Other aspects can be taken into account such as the communication latency that can be controlled by the depth of the tree.

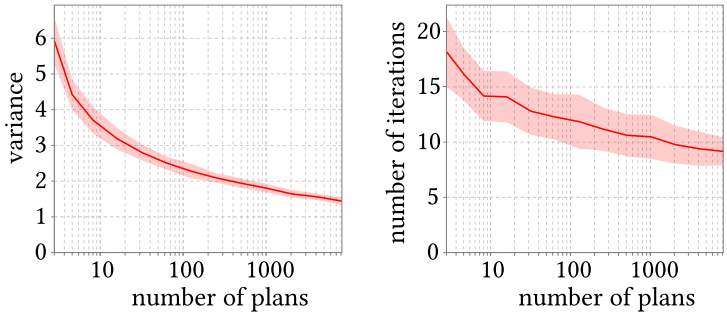
6.2 Number and Size of Plans

In Figure 6.7 of the earlier work on the EPOS optimization (Pournaras 2013), it is shown that an increasing number of possible plans achieves higher reduction in global cost. Varying the number of possible plans has a social influence and relevance as well: a very low number of plans may be perceived as a low level of system flexibility and therefore as overtake of human autonomy, which can result in low participation and satisfaction. However, a very high[7] number of possible plans may result in information overwhelming, low comprehension and feelings of regret as earlier shown in real-world applications of discrete choice theory, for instance, privacy choices (Korff and Böhme 2014). This article revisits this critical feature of the computational problem by confirming whether the variance reduction and convergence speed increase in the case of the I-EPOS learning. Moreover, this section sheds light on the influence the size of the possible plans has on performance.

Figure 12(a) confirms the significant decrease of the global cost, i.e., variance reduction, under an increasing number of possible plans: $p \in \{2^1, 2^2, \ldots, 2^{13}\}$. The variance reduction is over 60% when the possible plans increase from 2 to 100. This is because of the larger degree of freedom to choose solutions with lower variance, given that the variance is bounded to zero. Along with this improvement, a striking finding is shown in Figure 12(b). Convergence speed significantly decreases as the number of possible plans increase.

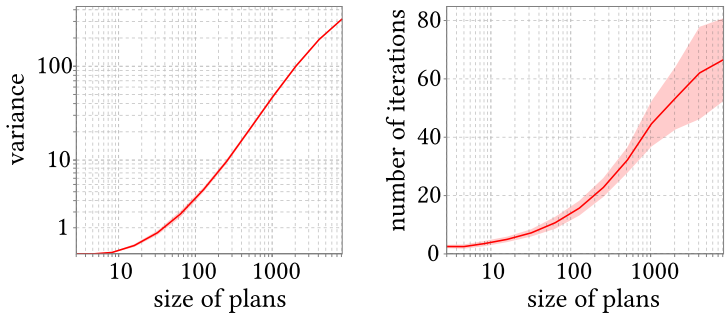
Earlier work shows that as data dimensionality increases, the distance of the data point in the closest proximity approaches the distance of the one in the farthest proximity, in other words, the distance between two randomly sampled data points is approximately equal (Beyer et al. 1999).

[7]Very high numbers of possible plans require an automated and intelligent processing by an information system rather than by humans. For instance, computing groups of possible plans according to features such as the average local cost is a way for citizens to manage a large number number of possible plans.



(a) Global cost (b) Convergence speed

Fig. 12. Performance evaluation for different numbers of plans p.



(a) Global cost (b) Convergence speed

Fig. 13. Performance evaluation for different plan size d.

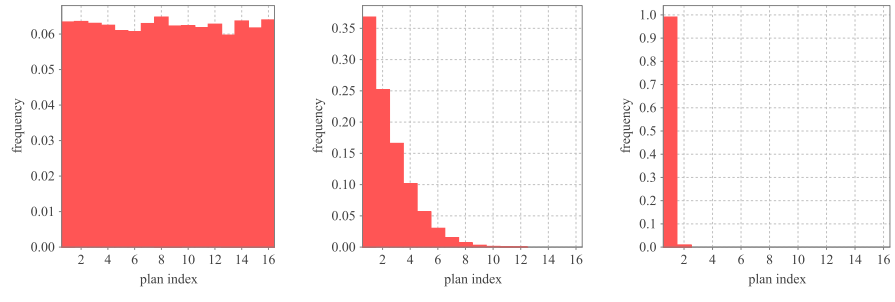
This finding has the following implications in the context of an increasing size of the possible plans: the impact of changing the selected plan to improve the variance reduction turns out to be insignificant, i.e., each possible plan result in approximately the same variance reduction.

This rationale is reflected in Figure 13. The size of the possible plans increases as $d \in \{2^1, 2^2, \ldots, 2^{13}\}$, while the variance steadily increases. The convergence speed also increases with a significant increase in standard deviation for higher plan sizes. Beyond this performance degrade, choosing a large size for the possible plans may turn out to be impractical: If the possible plans are time schedules, then uncertainties increase as the scheduling horizon increases. Moreover, a higher size of plans increases the communication, computational and storage cost in the agents.

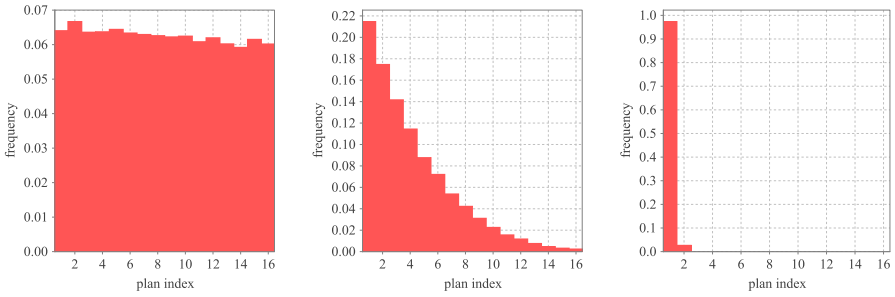
6.3 Agent Preferences

In this studied aspect, the agent selections of I-EPOS are biased by the local λ parameter that represents the agent preferences as shown in Equation (12). Despite the fact that the generated plans of the synthetic dataset are equivalent, i.e., equal mean values, for the purpose of this analysis it is assumed that the local agents' cost is the index of the selected plans and agents prefer plans with low index. The goal of this evaluation is to illustrate the performance tradeoffs by varying[8] the λ parameter as follows: (i) unbounded λ tested in the range $[0, 80]$ without the use of $(1 - \lambda)$ in Equation (12) and (ii) bounded λ in the range $[0, 1]$ according to Equation (12). When $\lambda = 1$ for $\lambda \in [0, 1]$, the global cost optimization is ignored and the minimization of the local cost is maximized

[8]The same λ value is set for all agents. Different λ values among agents is part of future work.

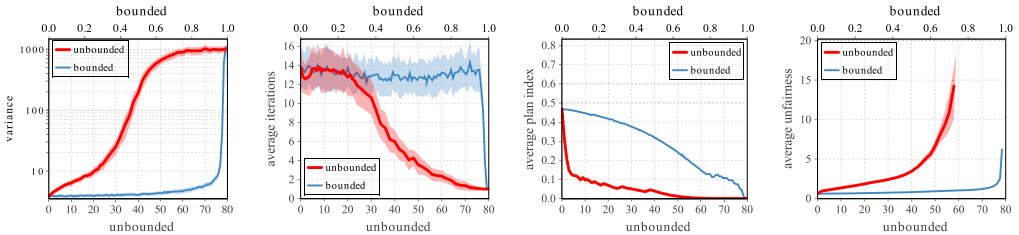


(a) Unbounded, $\lambda = 0.0$ (b) Unbounded, $\lambda = 5.0$ (c) Unbounded, $\lambda = 56.0$



(d) Bounded, $\lambda = 0.05$ (e) Bounded, $\lambda = 0.69$ (f) Bounded, $\lambda = 0.98$

Fig. 14. Histogram of the selected plan indices for different values of λ, unbounded $\lambda \in [0, \infty)$ and bounded $\lambda \in [0, 1]$.



(a) Global cost (b) Number of iterations (c) Average local cost (d) Unfairness

Fig. 15. Performance evaluation for different values of λ, unbounded $\lambda \in [0, 80]$ and bounded $\lambda \in [0, 1]$.

with all agents selecting the plan with index 0. Given the Gaussian distribution with which the synthetic dataset is generated, the $\lambda = 1$ is equivalent to agents making random selections, in terms of global cost.

Figure 14 illustrates the frequency of plan selections for various unbounded and bounded λ values. The shift of the distribution to low plan indices demonstrates the increasing algorithmic biases introduced towards these plans as the λ value increases.

Figure 15 illustrates the performance of I-EPOS for different unbounded and bounded λ values. The following observations can be made for the unbounded and bounded schemes of λ: Both schemes regulate the tradeoff of global vs. local cost. Higher λ values decrease local cost as shown in Figure 15(c), while global cost increases (Figure 15(a)). For unbounded λ, a range of possible λ values is determined empirically by searching for the range of λ values in which global and local

Table 3. Performance of the Standard Self-adaptive Plan Selection Scheme of I-EPOS Against the Gradiend Descent Schemes

Plan Selection Scheme	$E_G^{(1)}$	$E_G^{(t)}$	Convergence
Self-adaptive	**480 ± 70**	**3.2 ± 0.30**	**14 ± 2.3**
Adaptive gradient	**480 ± 70**	3.3 ± 0.27	**14 ± 2.3**
Local gradient	**480 ± 70**	3.3 ± 0.52	18 ± 3.4
Global gradient	**480 ± 70**	12 ± 1.9	15 ± 4.0

cost have high sensitivity. These ranges are [20, 50] for global cost in Figure 15(a) and [0, 10] for local cost in Figure 15(c). In contrast, the bounded λ has a fixed range by design, although global cost shows sensitivity for very high λ values (Figure 15(a)).

High λ values show faster convergence times as shown in Figure 15(b). Unfairness increases significantly for high λ values as Figure 15(d) shows. This is because of the unequal distribution of the selected plan indices for high λ values as depicted in Figure 14. The cutoff of unfairness values for the unbounded λ indicates the selection of a single plan index by all agents.

The results can be interpreted as follows: When individuals make choices in favor of their comfort (high λ values), collective efficiency is sacrificed in terms of global cost and fairness. Nevertheless, unbounded λ values around 10 to 20 or bounded values around 0.8 to 0.9 achieve a good tradeoff for individuals: very high reduction in local cost, while global cost remains low and fairness high.

6.4 Performance Comparison

This section compares the plan selection schemes of I-EPOS. Moreover, performance comparisons are shown with other state-of-the-art systems of decentralized combinatorial optimization.

Table 3 illustrates the comparison of the evaluated plan selection schemes. All approaches perform the initialization on the first iteration and therefore the global cost has the value of 480 in all schemes. However, the self-adaptive scheme has the lowest global cost and the highest convergence speed together with the adaptive gradient. The local gradient achieves lower global cost than the global gradient as it operates on replacing individual selections to make an improvement. These replacements cause the lowest convergence speed for the local gradient.

A fair comparison of I-EPOS with related work is not straightforward as the problem setup is highly challenging and there is a very limited number of algorithms designed to operate in a similar fashion as I-EPOS. Although several earlier algorithms and their applications draw parallels with the distributed design of I-EPOS, for instance, ant colony optimization for routing in wireless sensor networks (Ducatelle et al. 2005; GhasemAghaei et al. 2007), reinforcement learning for traffic light control (Dusparic and Cahill 2009), and load-balancing in cell tower of mobile networks (Hu et al. 2010), these algorithms are not directly applicable to the optimization problem illustrated in Section 2. For this reason, this section focuses on three state-of-the art algorithms and configurations capable of performing decentralized combinatorial optimization: (i) EPOS, (ii) COHDA, and (iii) Greedy. The rest of this section compares the design features of the algorithms and illustrates performance comparisons that underline the supreme performance of I-EPOS.

6.4.1 EPOS. EPOS (Pournaras 2013; Pournaras et al. 2017b) is an actual earlier design of I-EPOS that does not include learning capabilities and focuses entirely on optimization within a single bottom-up and top-down phase. EPOS and I-EPOS solve the same decentralized combinatorial optimization problems and have common domains of applicability (Pournaras et al. 2014a, 2014b).

Table 4. Performance Comparison of the Four Algorithms

Algorithm	Global cost computations		Vectors transmitted	
	Per agent	Critical path	Per agent	Critical path
I-EPOS	$O(pt)$	$O(pt \log a)$	$O(t)$	$O(t \log a)$
EPOS	$O(p^c)$	$O(p^c \log a)$	$O(p)$	$O(p \log a)$
COHDA	$O(pt)$	$O(pt)$	$O(at)$	$O(at)$
Greedy	$O(p)$	$O(ap)$	$O(1)$	$O(a)$

Their design though has a few significant differences, for instance, decision-making in EPOS takes place at the parents on behalf of the children, whereas, the decision-making in I-EPOS is fully localized. EPOS performs a non-local brute-force computation of all possible plan combinations of the children. This imposes certain computational constraints for tree topologies with a high number of children. Moreover, I-EPOS is capable of improving solutions dynamically, thanks to a fully decentralized iterative backpropagation mechanism. In contrast, EPOS operates in a single iteration and the top-down phase is an actual propagation of the global response computed.

6.4.2 COHDA. COHDA (Hinrichs et al. 2013, 2014) is an iterative asynchronous algorithm. In contrast to EPOS and I-EPOS, it does not rely on a tree topology for its operations. Because of this higher abstraction, the agents of COHDA incrementally exchange and merge with their neighbors complete sets of selected plans,[9] referred to in COHDA as the *knowledge base*, in contrast to EPOS and I-EPOS, respectively, that only exchange local and aggregated plans. A complete exchange of information is unscalable with the increase of network size and has a significant communication overhead in resource-constraint networks. For the purpose of the performance comparison, COHDA is configured to run over a tree topology.

6.4.3 Greedy. Greedy is a particular configuration of I-EPOS running for one iteration with at most one child per agent, i.e., agents in a sequence. This particular configuration corresponds to a sequential greedy optimization algorithm.

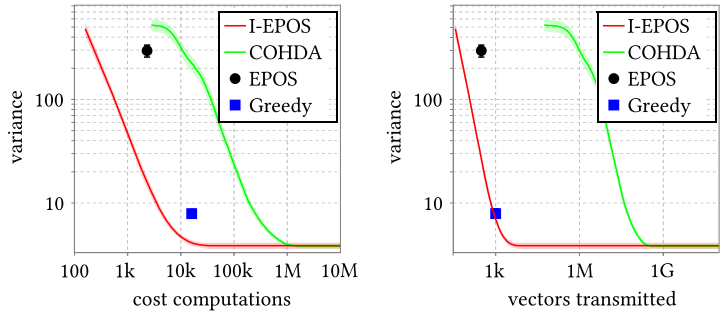
6.4.4 Evaluation. The three algorithms are compared with respect to the following two metrics: (i) *computational* and (ii) *communication* overhead. The former is computed by the number of global cost computations performed. The latter[10] measures the amount of data transmitted in the network and it is computed by the number and size of plans exchanged. These metrics can provide performance benchmarks and indicate the suitability of each algorithm for networks with scarce processing or energy resources, i.e., Internet of Things. Table 4 compares the performance of the four algorithms in respect to both metrics.

Performance is given for *each agent* and the *critical path* defined by the required sequence of agent executions. For instance, in the bottom-up phase of I-EPOS, the critical path corresponds to the tree height that is logarithmic to a. The shorter the critical path, the faster the algorithm execution is.

I-EPOS, COHDA, and Greedy perform local plan selection and therefore the computational complexity is linear to the number of plans p. EPOS has a higher computational load to perform given the combinational selections it performs. The computational complexity of I-EPOS and COHDA depends on the number of iterations executed. The computational complexity over the critical path

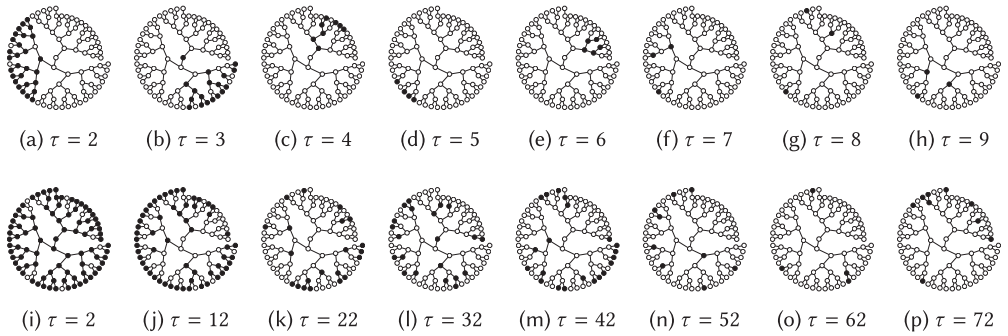
[9]COHDA uses counters for each agent selection to distinguish the most recent one.

[10]The communication overhead for building and maintenance of the tree or another dynamic topology for COHDA is not counted in these measurements as it is out of the scope of this work and subject of the network reliability and the application domain.



(a) Computational overhead. (b) Communication overhead.

Fig. 16. Performance comparison of the four algorithms over the critical path.



(a) $\tau = 2$ (b) $\tau = 3$ (c) $\tau = 4$ (d) $\tau = 5$ (e) $\tau = 6$ (f) $\tau = 7$ (g) $\tau = 8$ (h) $\tau = 9$

(i) $\tau = 2$ (j) $\tau = 12$ (k) $\tau = 22$ (l) $\tau = 32$ (m) $\tau = 42$ (n) $\tau = 52$ (o) $\tau = 62$ (p) $\tau = 72$

Fig. 17. ((a)–(h)) I-EPOS vs. ((i)–(p)) COHDA. Network snapshots showing the agents (in black) changing their selection and agents (in white) that do not change their selection.

depends on the network size. For I-EPOS and EPOS, it is the tree height that influences the computational complexity and it is logarithmic to the number of agents a. Agent selections in COHDA require an equal or higher number of iterations t than the tree height. The Greedy algorithm is computationally more expensive than the other algorithms given its sequential execution. I-EPOS and Greedy transmit a constant amount of data per agent and iteration. In contrast, the transmitted data of EPOS depend on the number of plans sent to the parent. COHDA has in principle the highest communication overhead by scaling linearly to the network size given that messages contain all agent selections.

Figure 16 illustrates the performance comparison on the critical path for the four algorithms. Tradeoffs of cost-effectiveness are illustrated by showing the resulting overhead for different levels of variance reduction. The consecutive iterations are the ones that increase the computational and communication overhead in Figure 16. This shows how fast I-EPOS and COHDA achieve a certain performance level, i.e., a variance reduction. I-EPOS outperforms all algorithms and shows a striking cost-effectiveness for a decentralized learning algorithm. After several iterations, meaning an invested computational and communication cost, the two algorithms converge at the same variance level. Results for each agent show a similar trend to the ones of the critical path except Greedy that is fully outlined in Table 4.

While I-EPOS and COHDA eventually converge to a similar performance level, the traversal of the optimization space varies significantly. Figure 17 visualizes this algorithmic effect by depicting which agents change their plan selection over runtime. For a clearer visual illustration,

Table 5. Performance Comparison of I-EPOS with Local and Combinational Plan Selections

Plan selections	$E_G^{(1)}$	$E_G^{(t)}$	Convergence
Local	480 ± 70	**3.2 ± 0.30**	14 ± 2.3
Combinational	**300 ± 39**	3.3 ± 0.27	**12 ± 2.3**

an experiment with 100 agents is shown. It is worth noticing that while I-EPOS outperforms COHDA by finding optimal solutions faster, it also performs a significantly lower number of changes in plan selection (190 in total). In COHDA all changes of plan selections are propagated to the neighbors. In contrast, changes in I-EPOS are performed in branches that over the passage of iterations get rapidly shorter. At the end, only a few single isolated changes in the selected plans maximize performance.

6.4.5 Local vs. Combinational Selections. Compared to the localized selections of I-EPOS, EPOS performs combinational selections by letting parents aggregate and sum up all combinations of possible plans generated by children. Combinational selections are applied to I-EPOS to compare the reduction of the global cost with localized selections. Table 5 summarizes the results.

The global cost in the first iteration as well as the convergence speed are superior for the combinational plan selections. This justifies the initial design of EPOS, which operates in a single iteration. However, the local plan selections eventually achieve a lower global cost on convergence.

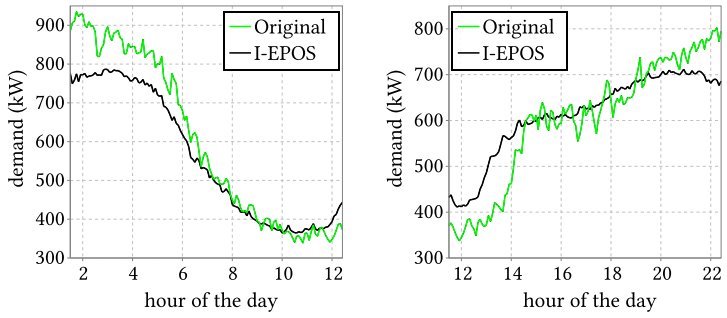
6.5 Application Scenarios on Sharing Economies

I-EPOS is applicable in the broader context of large-scale multi-agent systems. This article shows the broad and significant impact of the proposed generic algorithm on two very different scenarios of participatory sharing economies in the context of smart grids and smart cities: (i) *energy management* and (ii) *bike sharing*. The evaluation uses real-world data from state-of-the-art smart grid and smart cities pilot projects.

6.5.1 Energy Management. This application scenario envisions a highly participatory demand-response program for increasing system reliability by, for instance, preventing power peaks that can cause high energy costs and catastrophic blackouts (Pournaras et al. 2017a; Pournaras and Espejo-Uribe 2017). Residential consumers participate by equipping one or more controllable household appliances, e.g., refrigerators, water heaters, heating/cooling systems, and so on, with software that can operate the appliance according to plans selected by I-EPOS. Technology for this control level is feasible as discussed in earlier work (Kailas et al. 2012). Each household is represented by an I-EPOS agent that generates possible demand plans representing comfort and lifestyle flexibility, for instance, different times of taking a shower, or varied levels of thermostat setpoints. The global response corresponds to the total energy demand of all households aggregated. One way to reduce power peaks is to stabilize the demand by distributing it uniformly over time. This can be formalized as minimizing the variance of the global response. The variance is therefore used as the global cost function.

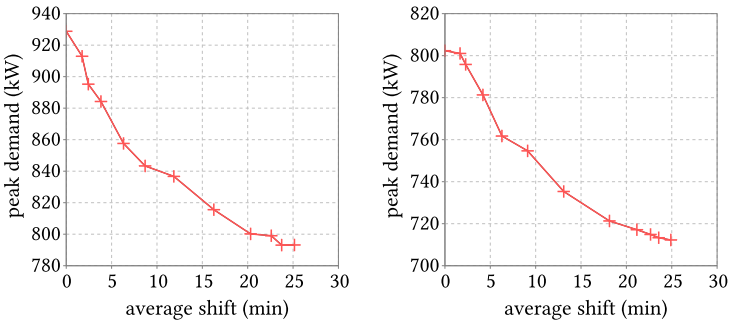
Real-world data from the Pacific Northwest Smart Grid Demonstration Project (PNW) by Battelle[11] are used for the experimental evaluation. The data contain 5-minute electricity consumption measurements from 1,000 residential households on 23.07.2014. Two plan generations are performed within the day and therefore the dataset is split in two parts, the PNW-MORNING for the duration 01:00–13:00 and the PNW-EVENING for 11:00–23:00, respectively. The cutoff

[11] Available on request at http://www.pnwsmartgrid.org/participants.asp (last accessed: September 2018).



(a) PNW-MORNING. (b) PNW-EVENING.

Fig. 18. Power peak-shaving by I-EPOS on the PNW dataset.



(a) PNW-MORNING (b) PNW-EVENING

Fig. 19. Tradeoff between peak demand reduction and average demand shift in the selected plans $\lambda \in \{0, 0.5, \ldots, 4, 5, 6, 100\}$ after $t = 60$ iterations for the PNW datasets.

duration is reserved for plan generation. A set of 13 possible plans is generated as follows: The measured demand is the first plan; the other 12 plans are generated by shifting the measured demand $5, 10, \ldots, 25$ or 30 minutes into the past or into the future.[12] The local cost of each plan is the amount of minutes shifted compared to the original demand.

The original power demand vs. I-EPOS global response for the PNW-MORNING and PNW-EVENING are illustrated in Figure 18. I-EPOS[13] reduces power peaks from 935 ± 0 to 792 ± 2 for PNW-MORNING and from 802 ± 0 to 713 ± 2 for PNW-EVENING, resulting in lower high-peak costs and instabilities in the power grid.

The peak-shaving capability of I-EPOS is also evaluated under preferences over the plans given that the possible plans are generated by a varied amount of shift. Figure 19 shows the tradeoff between peak demand reduction and the amount of shift in the selected plans averaged over all agents. The tradeoff is empirically controlled via the unbounded λ parameter. A $\lambda = 0$ corresponds to the default I-EPOS optimization without plan preferences and $\lambda = 100$ to the original demand.

6.5.2 Bike Sharing. In the context of smart cities, bike sharing is an important asset for improving urban qualities as citizens can use environmental friendly means of transportation, improve

[12]For example, the plan shifted 30 minutes into the future is the measured demand for the duration 01:00–12:00.
[13]The algorithm terminates after 26 ± 3.6 and 38 ± 4.1 iterations according to the second termination criterion of Section 4.4.

their individual health and decrease traffic congestion in densely populated cities. At the same time, they do not have to use their own bikes that can challenge the available parking spaces and increase the risk of stealing.

The broad establishment of bike sharing requires a high quality of service and low operational costs by making sure that citizens can always pick up a bike in a station and can always return it back to another one without the station exceeding the capacity of parked bikes. In other words, station should remain load-balanced under various conditions, such as population density, mobility, weather, and so on. Manual relocation of bikes by system operators is not viable in the long term and can increase operational costs significantly.

In the context of bike sharing, the possible plans may concern citizen recommendations about the stations from which bikes are picked up and to which they are returned. The possible plans are encoded as a vector with values the incoming minus the outgoing bikes of a citizen in each station at a certain time slot. For example, a citizen traveling from station 1 to station 3 and from station 4 to station 3 has the following plan: $(-1, 0, 2, -1, \dots)$. I-EPOS can select recommended stations for each citizen's agent[14] such that the number of bikes among the stations remains balanced. This can be formalized as minimizing the variance of the global response. The possible plans represent the utilization of the stations by each citizen in contrast to the energy domain in which load-balancing is performed over possible plans containing the residential energy demand over time.

I-EPOS generates bike sharing plans by reasoning based on real-world historical data[15] from the Hubway bike sharing system in Paris. Although this dataset does not contain personalized records, citizen trips are extracted from citizen information: zip-code, year of birth, and gender. All trips that have common values in these fields are assumed to be made by the same citizen. A random subset of 1,000 unique citizens represents the agents in I-EPOS, with a different seed for each run of the algorithm. The timeslot is chosen from 8:00am to 10:00am. All historic unique trips[16] a citizen did in the defined timeslot of a week day are considered as the possible plans for that day. The distance of the stations is encoded in the trips of the citizens. The local cost of each plan is defined by the likelihood that the citizen does not make the trip instructed in the plan. For instance, if three plans are chosen 4, 5 and 1 days of the measured time period respectively, the local cost for these plans is 0.6, 0.5, and 0.9, respectively.

Figure 20(a) illustrates the load-balancing of the stations using I-EPOS after 15 iterations. I-EPOS reduces the variance from roughly 230 to 0.58. This indicates a significant potential to reduce the number of manual bike relocations. However, recommendations may not be followed if the citizen is unlikely to choose a certain trip, i.e., a trip with high local cost. Figure 20(b) shows the tradeoff between global and local cost empirically controlled via the unbounded λ parameter after 30 iterations. A $\lambda = 0$ is the one extreme in which the local cost is not considered, in contrast to $\lambda = 100$ that results in a global response equivalent to the original data.

Results show that making plan selections with the average likelihood of the selected plan reduced in half is followed by a reduction of the variance by a factor of more than 100.

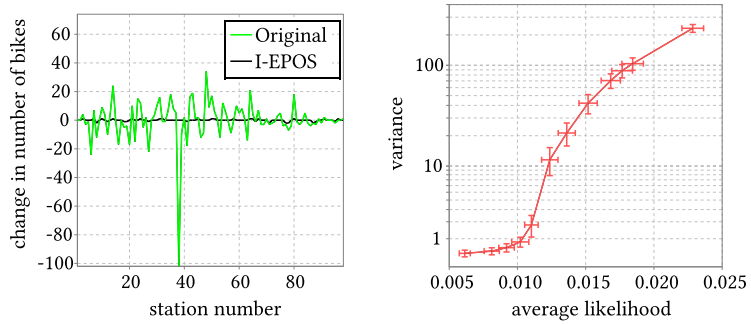
6.6 Global Optimality

The evaluation of global optimality is challenging for large-scale systems due to the exponential explosion $O(p^a)$ of the solution space. Two experiments are conducted with a feasible solution

[14] Such an agent can be implemented as a mobile app, for instance.

[15] The dataset is made available in the context of the Hubway Data Visualization Challenge: http://hubwaydatachallenge.org/ (last accessed: September 2018).

[16] As citizens do not travel each day, plans with no trips can be defined as well. These plans are not included in the dataset.



(a) Original bikes utilization vs. I-EPOS global response.

(b) Trade-off between variance and average likelihood of the selected plans $\lambda \in \{0, 1, \ldots, 10, 100\}$ after $t = 30$ iterations.

Fig. 20. I-EPOS performance for the bike sharing dataset.

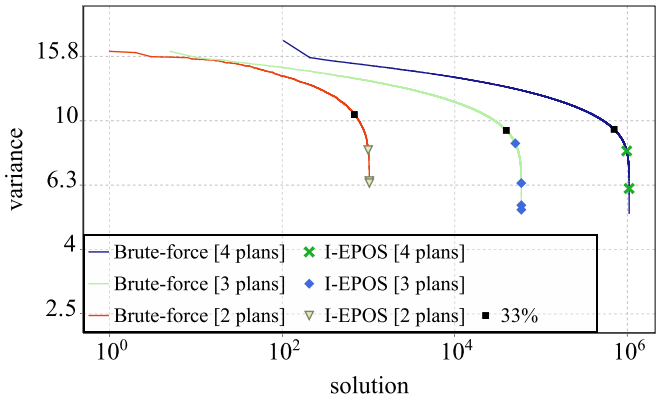
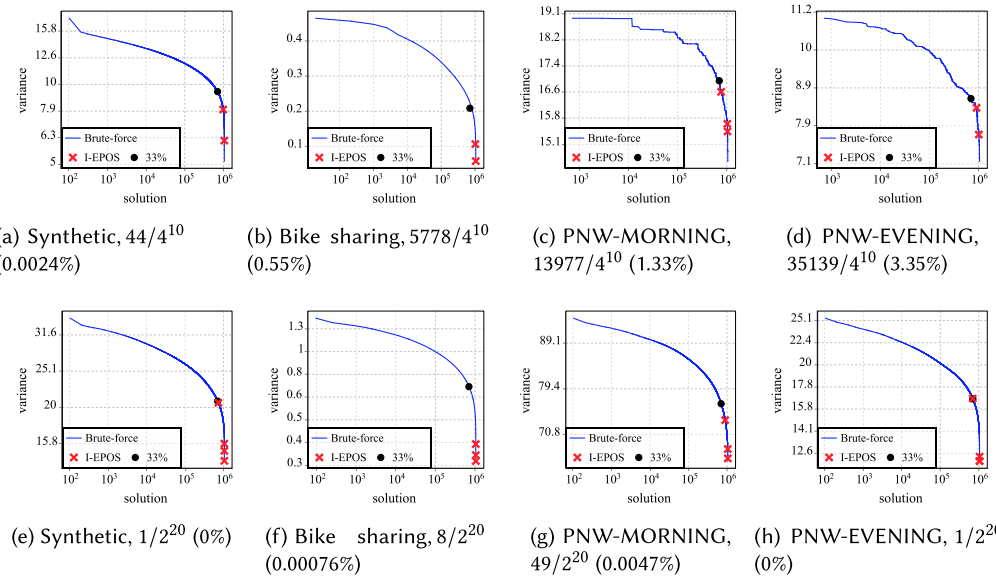


Fig. 21. I-EPOS vs. brute-force optimality for $a = 10$ and varying number of possible plans.

space computed by a brute-force search. In the first experiment, the synthetic dataset is used with a fixed number of agents $a = 10$, while the number of possible plans vary as $p = 2$, $p = 3$, and $p = 4$. The total number of combinations is $2^{10} = 1024$, $3^{10} = 59{,}049$ and $4^{10} = 1{,}048{,}576$ respectively. In the second experiment, a fixed solution space of size $p^a = 4^{10} = 2^{20} = 1{,}048{,}576$ is created, in which the number of agents and the number of possible plans vary such that the total number of combinations remains constant. The second experiment involves the synthetic, bike-sharing and energy datasets.

Figure 21 illustrates the results of the first experiment. The solutions are sorted according to the global cost, i.e., the variance. The projected ×, ⧫, ▽ points show the solutions found by I-EPOS during convergence and the ■ points show the top 33% of the solution space. An increase in the number of plans creates a solution space with solutions of higher as well as lower variance. This observation is in line with the results of Figure 12(a). In the case of $p = 2$ and $p = 3$, I-EPOS finds the optimum solution, while for $p = 4$ I-EPOS finds the 368th solution, which is approximately the top 0.035% of the solution space.

Figure 22 illustrates the second performed experiment. A striking finding is observed: the larger the number of agents, the better the solutions found are, despite the larger values of variance. This



(a) Synthetic, $44/4^{10}$ (0.0024%)
(b) Bike sharing, $5778/4^{10}$ (0.55%)
(c) PNW-MORNING, $13977/4^{10}$ (1.33%)
(d) PNW-EVENING, $35139/4^{10}$ (3.35%)
(e) Synthetic, $1/2^{20}$ (0%)
(f) Bike sharing, $8/2^{20}$ (0.00076%)
(g) PNW-MORNING, $49/2^{20}$ (0.0047%)
(h) PNW-EVENING, $1/2^{20}$ (0%)

Fig. 22. I-EPOS vs. brute-force optimality for different datasets. The numbers in parenthesis denote the top % of solutions found in the total solution space of size $p^a = 4^{10} = 2^{20} = 1{,}048{,}576$.

may now better explain Figure 10(a): Although variance increases for a low number of agents, this does not necessarily mean that the solutions found are worse. Instead, the variance cancellations when summing up the selected plans for a larger population of agents may explain this data artifact.

Another striking finding is the efficiency of I-EPOS to explore the optimization space from the very first iteration. The worst solution of I-EPOS during convergence, meaning the one found on the first iteration remains lower that the top 33% of all solutions and it is the one of PNW-EVENING with $a = 20$ and $p = 2$. In the first iteration, the level-by-level coordinated choices made in the bottom-up and top-down learning phase contribute significantly on the efficient exploration of the optimization space.

7 SUMMARY OF FINDINGS AND DISCUSSION

The key findings of the performed experiments are summarized as follows:

- A very few number of learning iteration, i.e., around 10 or fewer, are required for the convergence of I-EPOS. The first iterations contribute the highest to the reduction of global cost (Figure 9).
- An increase of a low number of agents increases global cost (Figure 10(a)) and lowers convergence speed (Figure 10(b)), while the global cost decreases by increasing further a high number of agents. An increase of the global cost is not related though with the optimality that actually improves for a higher number of agents (Figure 22).
- An increasing number of children in the tree topology decreases the global cost and increases the convergence speed, though to a very low extent (Figure 11). Exception is the organization of the network in a sequence, i.e., one child, that shows very high convergence speed and higher global cost.
- A larger number of plans decreases the global cost (Figure 12(a) and 21) and increases convergence speed (Figure 12(b)).

- Increasing the plans size increases global cost and decreases convergence speed (Figure 13).
- The λ parameter controls the tradeoff between global cost, average local cost, convergence speed, and fairness. Increasing λ results in lower average local cost, faster convergence, higher global cost and lower fairness (Figure 15, 19, and 20(b)).
- The self-adaptive plan selection scheme outperforms all other ones based on gradient descent.
- Strikingly I-EPOS outperforms other systems in both computational and communication cost, even if these systems perform heavy brute force operations or exchange the complete information (Figure 16).
- Combinational plan selections achieve better solutions than local selections in the first iterations, while local selections converge eventually to solutions with a lower global cost (Table 6.4.5).
- I-EPOS can effectively perform (i) peak-shaving of energy demand (Figure 18) to improve the reliability of smart grids and (ii) load-balancing of bike-sharing stations (Figure 20(a)) to minimize operational costs.
- The global optimality of I-EPOS in the solution spaces of size $p^a = 4^{10} = 2^{20} = 1{,}048{,}576$ for different datasets is the following: (i) below the top 33% of all solutions in the first learning iteration and (ii) below the top 3.35% of all solutions in the last learning iteration (Figure 22).

On the one hand, the monotonously decreasing learning curves along with the fast convergence of I-EPOS suggest that a supportive mechanism or an enhancement for the better exploration of the solution space has the potential to improve optimality even further. However, the design of such an improvement without sacrificing communication cost and convergence speed is highly challenging as also shown in the plan selection schemes based on gradient descent.

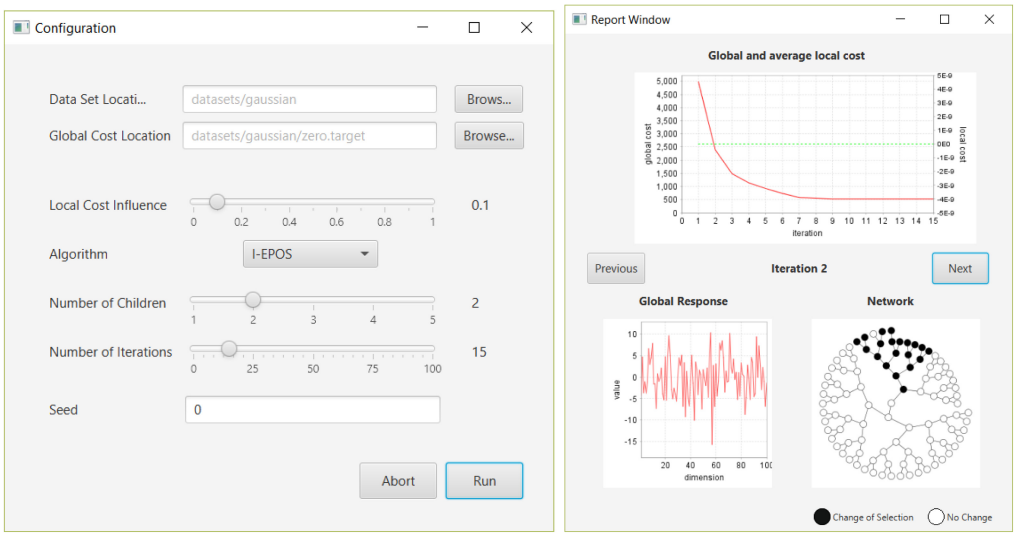
Linking the design of the learning algorithm with the design of a self-organizing tree network that adapts the topology as the means to discover better solutions seems a promising future line of research. Although the learning process of I-EPOS relies so far on a tree topology to perform the aggregation of the selected plans, more densely connected hierarchical structures as the ones of neural networks are worth further investigation. Furthermore, the impact of failures on the learning performance is out of the scope of this article but addressed in related and ongoing work (Pournaras et al. 2018).

Although I-EPOS relies on a universal cost function for agents that may have different roles (Bucchiarone et al. 2016) in a socio-technical system of sharing economies, the local λ parameter personalizes the agents' objectives by regulating the tradeoff of individual (local) vs. collective (global) criteria based on which the plan selections are performed. Reward mechanisms are means to incentivize agents for changing the choices of λ. The performance influence by the agents' positioning in the hierarchical network according to different criteria, for instance, according to the agents' λ value, is the subject of future evaluation.

8 I-EPOS AS A PARADIGMATIC ARTIFACT

Section 6.5 confirms that solutions to decentralized combinatorial optimization problems have a tremendous potential to build more sustainable and resilient digital societies. Authors here move a step forward to contribute a paradigmatic software implementation of I-EPOS together with other supporting software relevant for the broader research communities of distributed systems, optimization, artificial intelligence, machine learning, autonomic computing, multi-agent systems, game theory and others. The contributed exemplar[17] is a generic and modular open source Java

[17] Available at http://epos-net.org/shared/I-EPOS.zip (last accessed: September 2018).



(a) Input (b) Output

Fig. 23. The graphical user interface of I-EPOS.

implementation[18] of I-EPOS that provides the following opportunities for system evaluations: (i) Several different global and local cost functions. (ii) Possible plans of the same as well as different application domains generated from real-world and synthetic datasets. (iii) Different network settings, such as varying the topological properties of the network. The exemplar is also accompanied by a tutorial[19] and high-quality videos for a visual comprehension of the self-adaptive learning process. The software suite comes with a simulation and live software environment, which is prototyped with the Protopeer distributed toolkit (Galuba et al. 2009) for deployment in real-world testbeds such as Planetlab.[20] The community can also make use of integrated plotting and graph visualization capabilities as in Figure 17. Finally, the software suite comes with a graphical user interface for interactive executions as shown in Figure 23.

The object-oriented implementation of I-EPOS allows a straightforward experimentation without any change in the core I-EPOS code (black box use). Well-documented interfaces provide a high level abstraction and modularity, while allowing customization in different system setups and application domains. This is achieved with the *inheritance design pattern* that enables easy prototyping of combinatorial algorithms and cost functions. The base class of an algorithm implementation is the `agent.Agent` that defines the set of possible plans and the selected plan. It also defines a global and local cost function as well as the functionality for remote distributed communication by maintaining a limited neighbors list. Subclasses have to implement an active and passive state that define the algorithmic operations and the reactions to different received messages respectively. Two such classes are the `agent.IeposAgent` and `agent.CohdaAgent` for the I-EPOS and COHDA algorithms. The base class of a cost function is the `func.CostFunction`. It defines a method that receives as input a plan and returns the computed cost. The computations are specified in the subclasses, for instance, the minimum variance `func.VarCostFunction` or the

[18] Available at https://github.com/epournaras/EPOS (last accessed: September 2018).
[19] Available at https://github.com/epournaras/EPOS-Manual (last accessed: September 2018).
[20] Available at https://www.planet-lab.org. The toolkit is successfully used in the Euler high-performance cluster of ETH Zurich: Available at https://scicomp.ethz.ch/wiki/Main_Page (last accessed: September 2018).

`func.SqrDistCostFunction` cost function that minimizes the squared Euclidean distance from a target incentive signal (Pournaras et al. 2014a, 2014b). Logging follows the *observer design pattern* with the abstract class `agent.logging.AgentLogger` corresponding to the observer. It defines and writes to logs a serializable object containing the state of an agent. The logged information is defined in the subclasses. After each iteration, an agent sends its state to all its observers that handle the logging.

Although the contributed artifact is still a research prototype, several target groups can make an effective use of it. System developers guided by the contributed tutorials and interfaces can extend the artifact, design new optimization algorithms and use the implemented benchmarks for evaluation. In addition, policy-makers and non-computer scientist can interact with the software artifact via the graphical user interface to evaluate datasets and several system scenarios. Entrepreneurs can also use the I-EPOS artifact as a virtual laboratory of innovation by evaluating the feasibility of new application and business use-cases. Finally, the interplay of the scientific aspects of I-EPOS with art can provide new means for the general public to conceive decentralized collective learning processes that are too complex or non-intuitive for the mainstream thinking and general perception in society. Such work is the sonification of output data from I-EPOS to construct a constitutionally narrative of complex decentralized systems towards their equilibrium (Koutsomichalis and Pournaras 2017).

9 CONCLUSION AND FUTURE WORK

This article concludes that the decentralized collective learning approach of I-EPOS for multi-agent combinatorial optimization problems is feasible and can even significantly outperform related algorithms that either make use of non-local brute-force operations or exchange full information. The hierarchical structure over which an unsupervised learning is performed preserves by design autonomy and privacy, while it allows informational self-determination, active participation and a fully decentralized operation. These system properties enable new novel disruptive designs for participatory sharing economies in the context of smart grids and smart cities such as energy self-management or bike sharing initiatives. Experimental evaluation using real-world data from two state-of-the-art pilot projects in these domains provide a proof-of-concept for the broad applicability of I-EPOS. A software implementation of I-EPOS as an exemplar aims at settling a milestone for further work on collective learning and combinatorial optimization.

Future work includes the design of self-organizing hierarchical structures as the means to improve the learning performance. The introduction of agents' mobility (Bosse 2017) for more robust live deployments in the Internet of Things as well as an empirical performance analysis of agents residing on low power devices are part of future work. The evaluation of equilibria originated from reward mechanisms that incentivize the agent preferences is ongoing work. Finally, further applications and real-world prototypes such as charging coordination of electrical vehicles as well as mobile apps for scheduling of citizens' activities are in progress. Ultimately, the collective engagement of the broader research communities with decentralized learning is imperative for the creation of a more viable, ethically designed and socially responsible artificial intelligence.

APPENDIX

A COST FUNCTIONS

The cost functions studied in this article receive as input a d-dimensional vector denoted as $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and their output is a scalar denoting the cost of vector $\mathbf{x}$.

A quadratic cost function is non-linear and consists of a quadratic, a linear and a constant term. The quadratic term weights correlations between different dimensions i and j with the quadratic

Table 6. The Cost Functions Considered in This Paper

Cost function	Definition	Gradient
Quadratic	$f_{\text{quad}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{x}^\top \mathbf{L} + C$	$\nabla f_{\text{quad}}(\mathbf{x}) = (\mathbf{Q} + \mathbf{Q}^\top)\mathbf{x} + \mathbf{L}$
Linear	$f_{\text{lin}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{L} + C$	$\nabla f_{\text{lin}}(\mathbf{x}) = \mathbf{L}$
Variance	$f_{\text{var}}(\mathbf{x}) = \mathbf{x}^\top \tilde{\mathbf{Q}}\mathbf{x}$	$\nabla f_{\text{var}}(\mathbf{x}) = 2\tilde{\mathbf{Q}}\mathbf{x}$

$\mathbf{Q} \in \mathbb{R}^{d \times d}$ quadratic factors.
$\mathbf{L} \in \mathbb{R}^{d}$ linear factors.
$C \in \mathbb{R}$ constant scalar.
$\tilde{\mathbf{Q}} = \frac{1}{d-1}(\mathbf{I} - (\frac{1}{d})_{d \times d})$ quadratic factors of the variance cost function.

factors q_{ij}. The linear term weights the values of each dimension with the linear factors l_i. The constant term C is independent of the input to the cost function. The resulting cost function can be formalized as follows:

$$f_{\text{quad}}(\mathbf{x}) = \sum_{i=1}^{d} \sum_{j=1}^{d} q_{ij} x_i x_j + \sum_{i=1}^{d} l_i x_i + C. \tag{28}$$

The variance cost function computes the empirical variance of the input vector with each dimension representing one sample. The typical definition for the empirical variance is given as follows:

$$f_{\text{var}}(\mathbf{x}) = \frac{1}{d-1} \sum_{i=1}^{d} (x_i - \bar{x})^2, \quad \bar{x} = \frac{1}{d} \sum_{j=1}^{d} x_j. \tag{29}$$

Table 6 illustrates the formalism of the cost functions considered in this article. The definition of the cost functions is reformulated using a vector notation as well as the gradient definitions.

REFERENCES

Mohammad Alaggan, Sébastien Gambs, and Anne-Marie Kermarrec. 2011. *Private Similarity Computation in Distributed Systems: From Cryptography to Differential Privacy*. Springer, Berlin, 357–377.

Zeyuan Allen-Zhu and Elad Hazan. 2016. Variance reduction for faster non-convex optimization. In *Proceedings of the International Conference on Machine Learning*. 699–707.

Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. 1999. When is "nearest neighbor" meaningful? In *Proceedings of the International Conference on Database Theory*. Springer, 217–235.

Stefan Bosse. 2017. Incremental distributed learning with javascript agents for earthquake and disaster monitoring. *Int. J. Distrib. Syst. Technol.* 8, 4 (2017), 34–53.

Arthur Earl Bryson and Yu-Chi Ho. 1969. *Applied Optimal Control: Optimization, Estimation and Control*. Xerox College Publishing.

Antonio Bucchiarone, Martina De Sanctis, and Annapaola Marconi. 2016. Decentralized dynamic adaptation for service-based collective adaptive systems. In *Proceedings of the International Conference on Service-Oriented Computing*. Springer, Berlin, 5–20.

Javier Cámara, David Garlan, Bradley Schmerl, and Ashutosh Pandey. 2015. Optimal planning for architecture-based self-adaptation via model checking of stochastic games. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC'15)*. ACM, New York, NY, 428–435.

Ernest J. J. H. H. Chang. 1982. Echo algorithms: Depth parallel operations on general graphs. *IEEE Trans. Softw. Eng.* 8, 4 (1982), 391.

Anton Chechetka and Katia Sycara. 2006. No-commitment branch and bound search for distributed constraint optimization. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM, 1427–1429.

Cyrille Médard de Chardon, Geoffrey Caruso, and Isabelle Thomas. 2016. Bike-share rebalancing strategies, patterns, and purpose. *J. Transport Geogr.* 55 (2016), 22–39.

Li Deng, Dong Yu, et al. 2014. Deep learning: Methods and applications. *Found. Trends Sign. Process.* 7, 3–4 (2014), 197–387.

Frederick Ducatelle, Gianni Di Caro, and Luca Maria Gambardella. 2005. Using ant agents to combine reactive and proactive strategies for routing in mobile ad-hoc networks. *Int. J. Comput. Intell. Appl.* 5, 2 (2005), 169–184.

Ivana Dusparic and Vinny Cahill. 2009. Distributed w-learning: Multi-policy optimization in self-organizing systems. In *Proceedings of the 2009 3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 20–29.

Judith Fröhling. 2017. *Abstract Flexibility Description for Virtual Power Plant Scheduling*. Ph.D. Dissertation. BIS der Universität Oldenburg.

Wojciech Galuba, Karl Aberer, Zoran Despotovic, and Wolfgang Kellerer. 2009. ProtoPeer: A P2P toolkit bridging the gap between simulation and live deployement. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 60.

Reza GhasemAghaei, Md Abdur Rahman, Wail Gueaieb, and Abdulmotaleb El Saddik. 2007. Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks. In *Proceedings of the 2007 IEEE Instrumentation & Measurement Technology Conference (IMTC'07)*. IEEE, 1–6.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, MA.

Sara Hajian, Francesco Bonchi, and Carlos Castillo. 2016. Algorithmic bias: From discrimination discovery to fairness-aware data mining. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2125–2126.

Dirk Helbing, Bruno S. Frey, Gerd Gigerenzer, Ernst Hafen, Michael Hagner, Yvonne Hofstetter, Jeroen van den Hoven, Roberto V. Zicari, and Andrej Zwitter. 2017. Will democracy survive big data and artificial intelligence. *Sci. Am.* 25 (2017).

Dirk Helbing and Evangelos Pournaras. 2015. Society: Build digital democracy. *Nature* 527 (2015), 33–34.

Christian Hinrichs, Sebastian Lehnhoff, and Michael Sonnenschein. 2013. COHDA: A combinatorial optimization heuristic for distributed agents. In *Proceedings of the International Conference on Agents and Artificial Intelligence*. Springer, 23–39.

Christian Hinrichs, Sebastian Lehnhoff, and Michael Sonnenschein. 2014. A decentralized heuristic for multiple-choice combinatorial optimization problems. In *Proceedings of the Annual Conference on Operations Research 2012*. Springer, 297–302.

Honglin Hu, Jian Zhang, Xiaoying Zheng, Yang Yang, and Ping Wu. 2010. Self-configuration and self-optimization for LTE networks. *IEEE Commun. Mag.* 48, 2 (2010), 94–100.

Aravind Kailas, Valentina Cecchi, and Arindam Mukherjee. 2012. A survey of communications and networking technologies for energy management in buildings and home automation. *J. Comput. Netw. Commun.* 2012 (2012).

Hongseok Kim, Young-Jin Kim, Kai Yang, and Marina Thottan. 2011. Cloud-based demand response for smart grid: Architecture and distributed algorithms. In *Proceedings of the 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm'11)*. IEEE, 398–403.

Stefan Korff and Rainer Böhme. 2014. Too much choice: End-user privacy decisions in the context of choice proliferation. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS'14)*. 69–87.

Marinos Koutsomichalis and Evangelos Pournaras. 2017. The sound of decentralization-sonifying computational intelligence in sharing economies. In *Proceedings of the 23rd International Symposium on Electronic Art (ISEA'17)*.

Yung-Ming Li, Yong Tan, and Prabuddha De. 2013. Self-organized formation and evolution of peer-to-peer networks. *INFORMS J. Comput.* 25, 3 (2013), 502–516.

Ashutosh Pandey, Gabriel A. Moreno, Javier Cámara, and David Garlan. 2016. Hybrid planning for decision making in self-adaptive systems. In *Proceedings of the 10th International Conference on Self-adaptive and Self-organizing Systems*. IEEE.

Payal V. Parmar, Shraddha B. Padhar, Shafika N. Patel, Niyatee I. Bhatt, and Rutvij H. Jhaveri. 2014. Survey of various homomorphic encryption algorithms and schemes. *Int. J. Comput. Appl.* 91, 8 (2014).

Adrian Petcu and Boi Faltings. 2005. A Scalable Method for Multiagent Constraint Optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 266–271.

Peter Pilgerstorfer and Evangelos Pournaras. 2017. Self-adaptive learning in decentralized combinatorial optimization: A design paradigm for sharing economies. In *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE Press, 54–64.

Evangelos Pournaras. 2013. *Multi-level Reconfigurable Self-organization in Overlay Services*. Ph.D. Dissertation. TU Delft, Delft University of Technology.

Evangelos Pournaras, Ben-Elias Brandt, Manish Thapa, Dinesh Acharya, Jose Espejo-Uribe, Mark Ballandies, and Dirk Helbing. 2017a. SFINA-simulation framework for intelligent network adaptations. *Simul. Model. Pract. Theory* 72 (2017), 34–50.

Evangelos Pournaras and Jose Espejo-Uribe. 2017. Self-repairable smart grids via online coordination of smart transformers. *IEEE Trans. Industr. Inf.* 13, 4 (2017), 1783–1793.

Evangelos Pournaras, Matteo Vasirani, Robert E. Kooij, and Karl Aberer. 2014a. Decentralized planning of energy demand for the management of robustness and discomfort. *IEEE Trans. Industr. Inf.* 10, 4 (2014), 2280–2289.

Evangelos Pournaras, Matteo Vasirani, Robert E. Kooij, and Karl Aberer. 2014b. Measuring and controlling unfairness in decentralized planning of energy demand. In *Proceedings of the 2014 IEEE International Energy Conference (ENERGYCON'14)*. IEEE, 1255–1262.

Evangelos Pournaras, Srivatsan Yadhunathan, and Ada Diaconescu. 2018. Holarchic Structures for Decentralized Deep Learning-A Performance Analysis. arXiv preprint arXiv:1805.02686 (2018). http://arxiv.org/abs/1805.02686

Evangelos Pournaras, Mark Yao, and Dirk Helbing. 2017b. Self-regulating supply–demand systems. *Fut. Gener. Comput. Syst.* 76 (2017), 73–91.

Jakob Puchinger and Günther R. Raidl. 2005. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Proceedings of the International Work-Conference on the Interplay Between Natural and Artificial Computation*. Springer, 41–53.

R. Tyrrell Rockafellar, Stanislav Uryasev, et al. 2000. Optimization of conditional value-at-risk. *J. Risk* 2 (2000), 21–42.

Ognjen Scekic, Hong-Linh Truong, and Schahram Dustdar. 2013. Incentives and rewarding in social computing. *Commun. ACM* 56, 6 (2013), 72–82.

Thure Traber and Claudia Kemfert. 2011. Gone with the wind? - Electricity market prices and incentives to invest in thermal power plants under increasing wind energy supply. *Energy Econ.* 33, 2 (2011), 249–256.

William Yeoh, Ariel Felner, and Sven Koenig. 2008. BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, Vol. 2. International Foundation for Autonomous Agents and Multiagent Systems, 591–598.

Received February 2018; revised June 2018; accepted September 2018