Self-adaptive Learning in Decentralized Combinatorial Optimization – A Design Paradigm for Sharing Economies

Peter Pilgerstorfer and Evangelos Pournaras
Professorship of Computational Social Science
ETH Zurich,
Zurich, Switzerland
Email: {peterp,epournaras}@ethz.ch

***Abstract*—The democratization of Internet of Things and ubiquitous computing equips citizens with phenomenal new ways for online participation and decision-making in application domains of smart grids and smart cities. When agents autonomously self-determine the options from which they make choices, while these choices collectively have an overall system-wide impact, an optimal decision-making turns into a combinatorial optimization problem known to be NP-hard. This paper contributes a new generic self-adaptive learning algorithm for a fully decentralized combinatorial optimization: I-EPOS, the *Iterative Economic Planning and Optimized Selections*. In contrast to related algorithms that simply parallelize computations or big data and deep learning systems that often require personal data and overtake of control with implication on privacy-preservation and autonomy, I-EPOS relies on coordinated local decision-making via structured interactions over tree topologies that involve the exchange of entirely local and aggregated information. Strikingly, the cost-effectiveness of I-EPOS in regards to performance vs. computational and communication cost highly outperforms other related algorithms that involve non-local brute-force operations or exchange of full information. The algorithm is also evaluated using real-world data from two state-of-the-art pilot projects of participatory sharing economies: (i) *energy management* and (ii) *bicycle sharing*. The contribution of an I-EPOS open source software suite implemented as a paradigmatic artifact for community aspires to settle a knowledge exchange for the design of new algorithms and application scenarios of sharing economies towards highly participatory and sustainable digital societies.**

***Keywords*-learning; adaptation; optimization; decentralized system; network; sharing economy; smart grid; smart city**

I. Introduction

The pervasiveness of Internet of Things technologies and ubiquitous computing systems creates paramount opportunities for the establishment of 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 bicycle sharing. 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 [1], [2] or to balance the number of bicycles in stations [3]. In both cases, a large number of agents perform coordinated decision-making to collectively determine whether a power peak is reduced or whether bicycle stations have a bicycle to pick up and space to return one.

When autonomous agents have a set of self-determined options to choose from, 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 [4]. Most earlier work focuses on computational aspects and heuristics for efficiently computing solutions by splitting the computational problem into smaller pieces and parallelizing computations. Such approaches include branch and bound based algorithms, for instance, BnB-ADOPT [5], NCBB [6] and the dynamic programming approach of DPOP [7]. In contrast, this paper focuses on approximation heuristics of decentralized combinatorial optimization systems in which participatory agents locally self-determine their options from which they make choices. Optimization is performed in a fully decentralized fashion using coordinated remote interactions that orchestrate collective decision-making. In this highly challenging scope and problem setting, there is a very limited earlier work, mainly the EPOS [8], [9] and COHDA [10], [11] systems, which face significant scalability issues that limit their broader applicability, for instance EPOS performing low order but expensive non-local brute-force operations, while COHDA requiring a full information exchange between agents.

This paper introduces a novel, generic and highly efficient self-adaptive learning algorithm designed to solve fully decentralized combinatorial optimization problems: I-EPOS[1], the *Iterative Economic Planning and Optimized Selections*. I-EPOS combines decentralized optimization and self-adaptive learning and, in this sense, this dual capability opens up supreme opportunities for an alternative paradigmatic design to discriminatory big data profiling systems or centralized artificial intelligence (AI) systems that overtake control and violate agents' autonomy. This is also the actual motivation of this paper to study the I-EPOS applicability in two challenging application scenarios of participatory sharing economies: stretch the potential of the fully decentralized and participatory learning capability in I-EPOS to disrupt the norms of

[1]Available at http://epos-net.org (last accessed: March 2017)

centralized management in techno-socio-economic systems.

Agents in I-EPOS autonomously self-determine (i) possible plans that schedule the operation of an application and (ii) their preferences for these plans. The possible plans represent agents' flexibility. Agents are structured in self-organized tree topologies 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 improved solutions. Information exchange is always either local or aggregated. 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 thanks to the self-adaptation process over the tree topology, trade-offs between local vs. global costs as well as fairness are manageable and finally the cost-effectiveness is notoriously superior to related algorithms, where cost stands for computational and communication overhead.

The contributions of this paper are the following: (i) A new decentralized combinatorial optimization algorithm based on self-adaptive learning. (ii) A benchmark and performance comparison of I-EPOS with three other related algorithms that have not been systematically and rigorously compared in earlier work. (iii) The applicability of I-EPOS in two application scenarios of participatory sharing economies: energy management and bicycle sharing. Real-world data from state-of-the-art pilot projects are used for the experimental evaluation. (iv) The implementation of I-EPOS and other related algorithms as a paradigmatic artifact for promoting further research on decentralized learning and optimization as well as the design of new application scenarios.

This paper is organized as follows: Section II formulates the optimization problem and the challenges this paper tackles. Section III introduces I-EPOS and discusses its design aspects. Section IV experimentally evaluates I-EPOS, including a performance comparison with three other algorithms and the illustration of two application scenarios in participatory sharing economies. Section V discusses the implementation of I-EPOS as a paradigmatic artifact for community. Finally, Section VI concludes this paper and outlines future work.

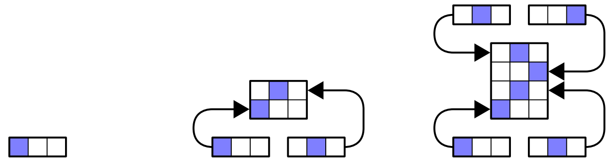
II. Decentralized Combinatorial Optimization

Table I summarizes the mathematical symbols used in this paper. Assume an *agent* $\mathfrak{a}$ with a finite set of *possible plans* $\mathcal{P}_\mathfrak{a}$ representing different operational schedules, for instance a time schedule for the allocation of resources, e.g. energy. A *plan* is a vector with real values about the allocation of resources. 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 1a shows the selected plan as one out of three possible plans. Plan generation can be performed with various methodologies that include clustering [12], Markov decision processes for fast and optimal plans [13], or model checking of stochastic multiplayer games [14].

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

TABLE I: 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 $\mathfrak{a}$
$\mathcal{C}_\mathfrak{a} \subset \mathcal{D}_\mathfrak{a}$	set of children for agent $\mathfrak{a}$
$\mathcal{D}_\mathfrak{a} \subset \mathcal{A}$	set of descendants for agent $\mathfrak{a}$
$\mathcal{O}(P, A) = \prod_{\mathfrak{a} \in A} P_\mathfrak{a}$	all combinations of sets $P_\mathfrak{a}$ for agents in A
$\mathcal{P}_\mathfrak{a} \subset \mathbb{R}^d$	possible plans of agent $\mathfrak{a}$
$a = \lvert\mathcal{A}\rvert$	number of agents
$c = \max_{\mathfrak{a} \in \mathcal{A}} \lvert\mathcal{C}_\mathfrak{a}\rvert$	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
$\mathbf{o} \in \mathcal{O}(P, A)$	a combination
$\mathbf{o}^\star \in \mathcal{O}(P, A)$	an optimal combination
$o_\mathfrak{a} \in \mathbf{o}$ or $\mathbf{o}_\mathfrak{a} \in \mathbf{o}$	plan of agent $\mathfrak{a}$ in combination $\mathbf{o} \in \mathcal{O}(P, A)$
$\mathfrak{a} \in \mathcal{A}$	an agent
$\mathfrak{c} \in \mathcal{C}_\mathfrak{a}$	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
$\tau \in \{1, ..., t\}$	number of the current iteration
$\mathbf{p} \in \mathcal{P}_\mathfrak{a}$	possible plan 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 $\mathbf{x}^{(\tau)} \in \{\mathbf{s}_\mathfrak{a}^{(\tau)}, \mathbf{g}^{(\tau)}, \mathbf{a}_\mathfrak{a}^{(\tau)}, \mathbf{t}_\mathfrak{a}^{(\tau)}\}$
$f_G : \mathbb{R}^d \rightarrow \mathbb{R}$	global cost function
$f_L : \mathbb{R}^d \rightarrow \mathbb{R}$	local cost function
$E_G^{(\tau)} = f_G\left(\mathbf{g}^{(\tau)}\right)$	global cost at iteration τ
$E_L^{(\tau)}$	average local cost at iteration τ
$U^{(\tau)}$	unfairness at iteration τ
$w : \mathbb{R}^d \rightarrow \mathbb{R}$	preference weight; raises the cost of disliked plans
$\lambda \in \mathbb{R}$	controls the trade-off between global and local cost
$\rho_{i,\mathfrak{a}} \in \mathbb{R}$	dislike of plan i by agent $\mathfrak{a}$



(a) Selected plan. (b) Aggregated response. (c) Global response.

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

summed up together form the *aggregated response* as shown in Figure 1b. The selected plans of all agents form a *global response* vector $\mathbf{g} = \sum_{\mathfrak{a} \in \mathcal{A}} \mathbf{s}_\mathfrak{a}$ shown in Figure 1c. 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 $\mathbf{o} \in \mathcal{O}(\mathcal{P}, \mathcal{A}) = \prod_{\mathfrak{a} \in \mathcal{A}} \mathcal{P}_\mathfrak{a}$ consists of one plan $\mathbf{o}_\mathfrak{a}$ per agent $\mathfrak{a}$, from which the optimal combination $\mathbf{o}^\star$ with the minimal global cost is selected. Cost minimization is defined as follows:

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

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. This means only small in size problems are feasible to solve optimally. 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 raises the need for approximation algorithms such as I-EPOS, the algorithm introduced in Section III.

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

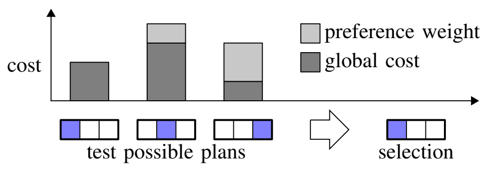


Fig. 2: Agent preferences in plan selection.

- Plan selections with a low average local cost E_L, so that agents have also an explicit local benefit from their plan selections. It is computed as follows:

$$E_L = \mu\{f_L(\mathbf{s}_\mathfrak{a}) \mid \mathfrak{a} \in \mathcal{A}\}. \tag{2}$$

- Plan selections, whose local cost has a low standard deviation among agents, so that agents' contributions are equally distributed. In this case the standard deviation is a measure of system fairness [15]. In this paper, the unfairness U is computed as the standard deviation normalized with the mean of the local cost for all selections:

$$U = \frac{\sigma\{f_L(\mathbf{s}_\mathfrak{a}) \mid \mathfrak{a} \in \mathcal{A}\}}{\mu\{f_L(\mathbf{s}_\mathfrak{a}) \mid \mathfrak{a} \in \mathcal{A}\}}. \tag{3}$$

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

III. Self-adaptive Learning

The network is assumed (self-)organized in a tree topology, as this structure can be constructed and maintained within a dynamic and distributed environment for arbitrary connected networks using AETOS [16], [9], [17] or the ECHO [18] algorithm, for example. A tree is a cycle-free connected network. 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, and therefore, a self-adaptive learning can be performed iteratively in a similar fashion as in the hierarchical structures of neural networks. The *root agent* is denoted as $\mathfrak{r}$. Each agent $\mathfrak{a}$ has a set of *children* $\mathcal{C}_\mathfrak{a}$ and a set of *descendants* $\mathcal{D}_\mathfrak{a}$, with $\mathcal{C}_\mathfrak{a} \subseteq \mathcal{D}_\mathfrak{a}$. 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' 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 $\tau = 1, \dots, t$ are explained below. To simplify the equations, the selected plans at iteration 0 are assumed to be zero: $\mathbf{s}_\mathfrak{a}^{(0)} = \mathbf{0}, \quad \forall \mathfrak{a} \in \mathcal{A}$.

```
Input: agent $\mathfrak{a}$, plans $\mathcal{P}_\mathfrak{a}$
Result: selected plan $\mathbf{s}_\mathfrak{a}^{(t)}$, aggregated response $\mathbf{a}_\mathfrak{a}^{(t)}$, global response $\mathbf{g}^{(t)}$
$\mathbf{s}_\mathfrak{a}^{(0)} \leftarrow \mathbf{0}, \quad \mathbf{a}_\mathfrak{a}^{(0)} \leftarrow \mathbf{0}, \quad \mathbf{g}^{(0)} \leftarrow \mathbf{0}, \quad \mathbf{t}_\mathfrak{c}^{(0)} \leftarrow \mathbf{0} \quad \forall \mathfrak{c} \in \mathcal{C}_\mathfrak{a}$;
for $\tau$=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
            $\tilde{\delta}_\mathfrak{c}^{(\tau)} \leftarrow 1, \quad \forall \mathfrak{c} \in \mathcal{C}_\mathfrak{a}$;
        else
            compute the preliminary deltas $\tilde{\delta}_\mathfrak{c}^{(\tau)}$ $\forall \mathfrak{c} \in \mathcal{C}_\mathfrak{a}$ from Equation 4 ;
        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 5;
    select preliminary plan $\tilde{\mathbf{s}}_\mathfrak{a}^{(\tau)}$ according to Equation 6;
    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
    $\delta_\mathfrak{c}^{(\tau)} \leftarrow \delta_\mathfrak{a}^{(\tau)} \tilde{\delta}_\mathfrak{c}^{(\tau)}, \quad \forall \mathfrak{c} \in \mathcal{C}_\mathfrak{a}$;
    send global response $\mathbf{g}^{(\tau)}$ and delta value $\delta_\mathfrak{c}^{(\tau)}$ to each child $\mathfrak{c} \in \mathcal{C}_\mathfrak{a}$;
    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 $\mathfrak{c} \in \mathcal{C}_\mathfrak{a}$ according to Equation 8;
end
```

Algorithm 1: The I-EPOS algorithm.

A. Bottom-up phase

In this phase, each agent has knowledge about the changes of the aggregated response performed by 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 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 receives the preliminary selections from its children and computes its own preliminary selection, preliminary aggregated response, preliminary global response and the preliminary delta values for its children.

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. This approach is referred to as MIA, the *Maximal Improvement in Aggregation*. An agent's knowledge about its children is illustrated in Figure 3.

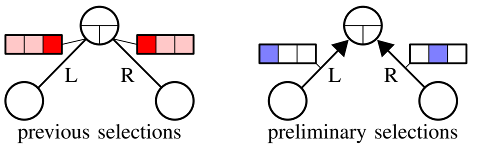


Fig. 3: The input of an agent by its children.

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

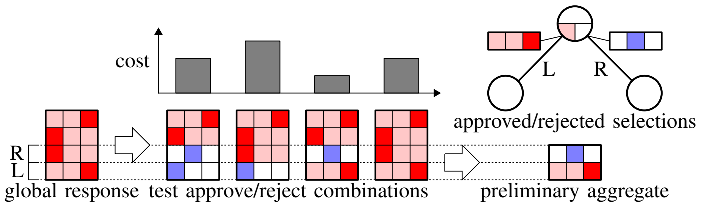

cost
L
R
approved/rejected selections
R
L
global response
test approve/reject combinations
preliminary aggregate

Fig. 4: Approval or rejection of branch selections with MIA.

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 in contrast to related work [11], [10] that relies on system-wide exchange of the actual selected plans instead of aggregated ones. At the first iteration when there is no history, all changes are approved. For all iterations after the first one, each agent evaluates all possible combinations of branch approvals and rejections $\mathcal{O}(\mathcal{B}, \mathcal{C}_{\mathfrak{a}})$ and selects a combination that maximally improves the global response. The delta value, that encodes a subtree approval or a subtree rejection, for child $\mathfrak{c}$ in combination $\mathbf{o} \in \mathcal{O}(\mathcal{B}, \mathcal{C}_{\mathfrak{a}})$ 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 $\mathbf{o}$ on the global response is evaluated as follows: $\mathbf{g}^{(\tau-1)} + \sum_{\mathfrak{c}\in\mathcal{C}_{\mathfrak{a}}} o_{\mathfrak{c}} \nabla\tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}$. A combination $\mathbf{o}^{\star}$ that results in the lowest global cost is selected[2]. The delta values of $\mathbf{o}^{\star}$ are chosen as the preliminary deltas $\tilde{\delta}_{\mathfrak{c}}^{(\tau)}$ of the respective child $\mathfrak{c}$. The delta value selection can be formalized as follows:

$$\mathbf{o}^{\star} = \underset{\mathbf{o}\in\mathcal{O}(\mathcal{B},\mathcal{C}_{\mathfrak{a}})}{\arg\min}\, f_G\left(\mathbf{g}^{(\tau-1)} + \sum_{\mathfrak{c}\in\mathcal{C}_{\mathfrak{a}}, \mathbf{o}_{\mathfrak{c}}\in\mathbf{o}} o_{\mathfrak{c}} \nabla\tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}\right)$$
$$\tilde{\delta}_{\mathfrak{c}}^{(\tau)} = \mathbf{o}_{\mathfrak{c}}^{\star}, \quad \forall \mathfrak{c} \in \mathcal{C}_{\mathfrak{a}}. \tag{4}$$

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:

$$\tilde{\mathbf{a}}_{\mathfrak{a}}^{(\tau)} = \mathbf{a}_{\mathfrak{a}}^{(\tau-1)} + \sum_{\mathfrak{c}\in\mathcal{C}_{\mathfrak{a}}} \tilde{\delta}_{\mathfrak{c}}^{(\tau)} \nabla\tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}$$
$$\tilde{\mathbf{g}}_{\mathfrak{a}}^{(\tau)} = \mathbf{g}^{(\tau-1)} + \sum_{\mathfrak{c}\in\mathcal{C}_{\mathfrak{a}}} \tilde{\delta}_{\mathfrak{c}}^{(\tau)} \nabla\tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}. \tag{5}$$

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.

2) Plan selection: The effective plan selections follow the maximum improvement design principle of MIA. The concept is illustrated in Figure 5.

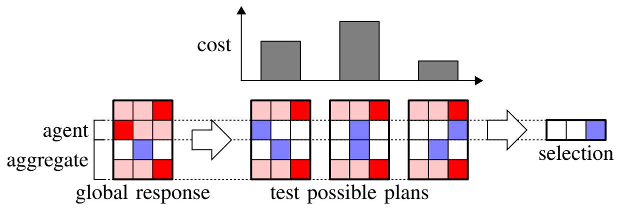


Fig. 5: Plan selection.

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:

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

where the preference weight $w(\mathbf{p}_{i,\mathfrak{a}})$ is normalized using the standard deviation of the different simulated global costs for

[2]If multiple combinations are optimal, $\mathbf{o}^{\star}$ is chosen uniformly at random out of all optimal combinations.

[3]If multiple plans are optimal, the selected plan is chosen uniformly at random out of all optimal plans.

the respective possible plans. The $\rho_{i,\mathfrak{a}}$ expresses the dislike of an agent $\mathfrak{a}$ towards a plan i.

3) Parent informing: Every non-root agent informs its parent about the selections of its respective branch with the preliminary selected plan and the preliminary aggregated plan of the agent. This procedure is shown in Figure 6.

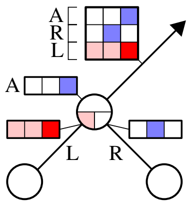


Fig. 6: Agent output during the bottom-up phase.

B. *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.

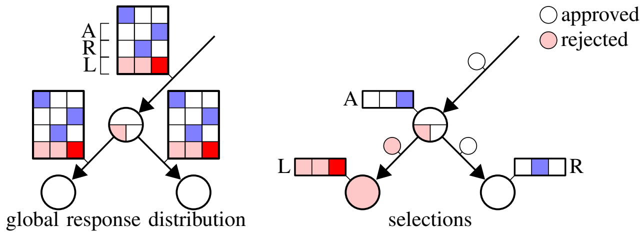
The root agent computes the global response based on its preliminary plans and responses that propagates downwards to all other agents in the network. For the root agent, the preliminary selected plan as well as the preliminary aggregated response correspond to the effective selections:

$$\mathbf{s}_{\mathfrak{r}}^{(\tau)} = \tilde{\mathbf{s}}_{\mathfrak{r}}^{(\tau)}$$
$$\mathbf{a}_{\mathfrak{r}}^{(\tau)} = \tilde{\mathbf{a}}_{\mathfrak{r}}^{(\tau)}. \quad (6)$$

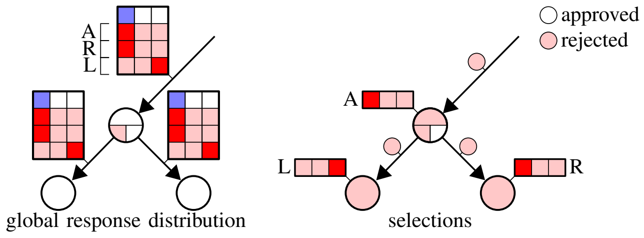
The preliminary deltas $\tilde{\delta}_{\mathfrak{c}}^{(\tau)}$ computed with MIA are used at this stage to determine the effective delta values $\delta_{\mathfrak{c}}^{(\tau)}$. The selection of the root agent is always approved, hence $\delta_{\mathfrak{r}}^{(\tau)} = 1$. The delta value for the other 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:

$$\delta_{\mathfrak{c}}^{(\tau)} = \delta_{\mathfrak{a}}^{(\tau)} \tilde{\delta}_{\mathfrak{c}}^{(\tau)}, \quad \forall \mathfrak{c} \in \mathcal{C}_{\mathfrak{a}}. \quad (7)$$

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, 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 aggregated responses are computed as follows:



(a) Ancestors send an approval signal.



(b) Ancestors send a rejection signal.

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

$$\mathbf{s}_{\mathfrak{a}}^{(\tau)} = \mathbf{s}_{\mathfrak{a}}^{(\tau-1)} + \delta_{\mathfrak{a}}^{(\tau)} \nabla \tilde{\mathbf{s}}_{\mathfrak{a}}^{(\tau)}$$
$$\mathbf{a}_{\mathfrak{a}}^{(\tau)} = \mathbf{a}_{\mathfrak{a}}^{(\tau-1)} + \delta_{\mathfrak{a}}^{(\tau)} \nabla \tilde{\mathbf{a}}_{\mathfrak{a}}^{(\tau)}$$
$$\mathbf{t}_{\mathfrak{c}}^{(\tau)} = \mathbf{t}_{\mathfrak{c}}^{(\tau-1)} + \delta_{\mathfrak{c}}^{(\tau)} \nabla \tilde{\mathbf{t}}_{\mathfrak{c}}^{(\tau)}, \quad \forall \mathfrak{c} \in \mathcal{C}_{\mathfrak{a}}. \quad (8)$$

C. *Other system design aspects*

Some other design aspects are discussed below: (i) autonomy, self-determination and participation, (ii) learning principle and monotonous improvement, (iv) termination.

1) Autonomy, self-determination and participation: The possible plans are self-determined by the agents as they are locally generated without I-EPOS been involved in this process. The algorithm does not impose (i) the plans, (ii) their number or even (iii) the overall participation into the process of generating plans at first place. I-EPOS can even operate with agents that only have a single possible plan, in other words, inflexible agents that acquire full control of their operation. Even in this extreme case, the self-adaptive learning process of I-EPOS is designed to optimize the system 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 relevant incentive systems [19] and reward mechanisms [20] .

2) Learning principle and monotonous improvement: In contrast to the earlier self-optimizing approach of EPOS [8], [9], the learning process of I-EPOS is self-adaptive as plan selections are a collective result computed within the bottom-up and top-down phase of a single iteration as well as across different iterations by using aggregated historical information.

The MIA mechanism of 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. MIA ensures that independently determined improvements of the global response by different tree branches do not result in decreased performance when aggregated.

To certain extent, the learnign concept draws parallels from backpropagation [21] in neural networks: In the forward pass (bottom-up phase) the agents predict their new selections and 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 plan 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 MIA either reduces the global cost or at least maintains the same global cost if the root agent selects the responses of the previous iteration.

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

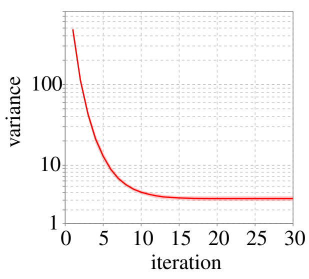
- *Termination if no agent performs a plan change*. This solution can be implemented with a boolean flag that agents can pass in the top-down phase. This termination approach may cut out some further improvements: agents may choose a different possible plan with equal cost[4] that may lead to new combinatorial solutions with further improvements in the next iterations. For the same reason, termination is not guaranteed as selections may change without improvement in the global cost.
- *Termination once the global cost equals the one of the previous iteration*. This approach is similar to previous one, 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 not chosen effectively or does not match the empirical data in use, it may lead to significant overhead as well.

IV. Experimental Evaluation

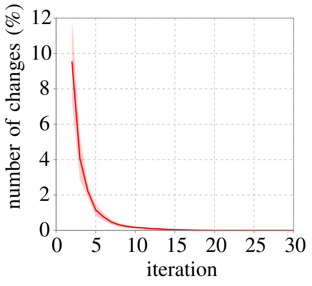
Experimental evaluation is performed in a network of 1000 agents randomly organized in height-balanced binary tree, i.e. a binary tree of minimal height. Each experiment is repeated 50 times with different seeds in the random number generators. Evaluation is performed with both (i) *synthetic* and (ii) *real-world* data from two application domains as illustrated in Section IV-B. The synthetic data concern 16 possible plans of size 100 generated from a standard normal distribution at every experiment repetition. By default, agents adopt $\lambda = 0$ 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.

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

Figure 8a 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.



(a) Global cost.



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

Fig. 8: Variance reduction and changes of selections in I-EPOS.

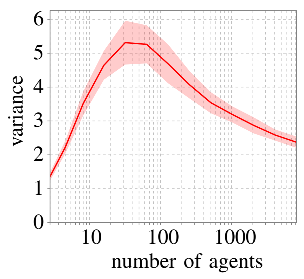
On average, the variance is reduced from 480 ± 70 in the initial iteration to 3.2 ± 0.30 after 30 iterations. As a scale of comparison: randomly choosing standard normally distributed plans results in an approximate variance of 1000 ± 140 on average[5]. The second termination criterion of Section III-C3 completes the algorithm in 14 ± 2.3 iterations.

Figure 9 evaluates two scalability aspects of I-EPOS: the influence of variance under increasing number of (i) agents and (ii) children. Figure 9a shows that a low number of agents results in a low number of combinations and therefore the variance increases as more plans are introduced. After a critical point, the combinations explode and therefore variance starts decreasing as the number of agents increases. Figure 9b confirms that the inter-agent communication within the tree structure can be optimized for network delays without a significant influence on the algorithm performance.

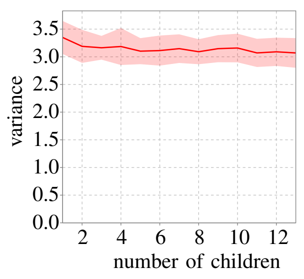
A. Performance comparison

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 [22], [23], reinforcement learning for traffic light control [24] and load-balancing in cell tower of mobile networks [25], these

[5]The sum of 1000 standard normally distributed plans is normally distributed with zero-mean and a variance of $\sigma^2 = 1000$. 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 1000 with a standard deviation of about 142.



(a) Increasing number of agents.



(b) Increasing number of children.

Fig. 9: The influence of variance under increasing number of agents and children.

algorithms are not directly applicable to the optimization problem illustrated in Section II. 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.

1) EPOS: EPOS [8], [9] is an actual earlier design of I-EPOS that does not include its learning capability and focus 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 [12], [15]. Their design though has a few significant differences, for instance, decision-making in EPOS takes place in 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.

2) COHDA: COHDA [10], [11] 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 nodes of COHDA incrementally exchange and merge with their neighbors complete sets of selected plans[6], 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 can cause a significant communication overhead in resource-constraint networks. For the purpose of the performance comparison, COHDA is configured to run over a tree topology.

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.

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 amount of global cost computations performed. The latter[7] measures the amount of data transmitted in the network and it is computed by the number of plans and response vectors exchanged. These metrics can provide performance benchmarks and indicate the suitability of each algorithm for Internet of Things networks and networks with scarce processing or energy resources. Table II compares the performance of the four algorithms in respect to both metrics.

TABLE II: 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	$O(p)$	$O(ap)$	$O(1)$	$O(a)$

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.

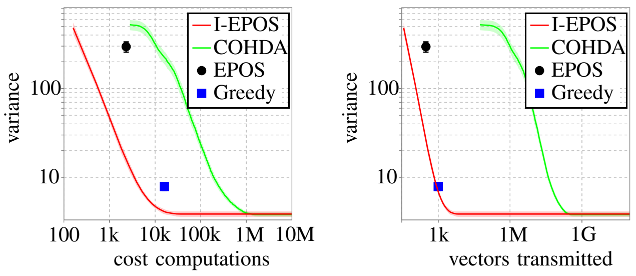
I-EPOS, COHDA and Greedy perform local plan selection and therefore the computational complexity is linear to the number of plans p. EPOS has a higher computational load to perform given the combinational selections it performs. The computational complexity of I-EPOS and COHDA depends on the number of iterations executed. The computational complexity over the critical path 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 10 illustrates the performance comparison on the critical path for the four algorithms. Trade-offs of cost-effectiveness are illustrated by showing the resulted overhead for different levels of variance reduction. The consecutive iterations are the ones that increase the computational and communication overhead in Figure 10. This shows for I-EPOS and COHDA how fast each algorithm achieves a certain performance level, i.e. a variance reduction. I-EPOS outperforms

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

[7]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 context of this work and it is subject of the network reliability and the application domain.

all algorithms and shows a paramount 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 II.



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

Fig. 10: Performance comparison of the four algorithms over the critical path.

While I-EPOS and COHDA eventually converge to a similar performance level, their traversal of the optimization space varies significantly. Figure 11 visualizes this effect of the algorithms by depicting which agents change their plan selection over runtime. For a clearer visual illustration, an experiment with 100 agents is shown. It is worth noticing that while I-EPOS overperforms 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 need to be propagated to the neighbors. In contrast, changes in EPOS are performed in branches that over the passage of the iterations get rapidly shorter. At the end, only a few single isolated changes in the selected plans contribute to a maximal performance.

B. Application scenarios on sharing economies

This paper shows the broad and significant impact of the proposed generic algorithm on two participatory sharing economy scenarios in smart grids and smart cities: (i) *energy management* and (ii) *bicycle sharing*. Although I-EPOS is applicable in the broader context of large-scale multi-agent systems, the two very different in nature applicability domains are critical and urgent for building a more sustainable society and they are chosen so that the generic design of I-EPOS is stretched. The evaluation uses real-world data from state-of-the-art smart grid and smart cities pilot projects.

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 [1], [2]. Residential consumers participate by equipping one or more controllable household appliances, e.g. refrigerators, water heaters, heating/cooling systems etc., 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 [26]. 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[8] are used for the experimental evaluation. The data contain 5-minute electricity consumption measurements from 1000 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 cut off duration is used 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[9]. 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 12. I-EPOS[10] reduces power peaks from 940 ± 0 to 790 ± 2 for PNW-MORNING and from 800 ± 0 to 710 ± 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 plan preferences given that the possible plans are generated by a varied amount of shift. Figure 13 shows the trade-off between peak demand reduction and the amount of shift in the selected plans averaged over all agents. The trade-off is controlled via the λ parameter. A $\lambda = 0$ corresponds to the default I-EPOS optimization without plan preferences and $\lambda = 100$ to the original demand.

2) Bicycle sharing: In the context of smart cities, bicycle sharing is an important asset for improving urban qualities as citizens can use environmental friendly means of transportation, improve their individual health and decrease traffic congestion in densely populated cities. At the same time, they do not have to use their own bicycles that can challenge the available parking spaces and increase the risk of stealing.

The broad establishment of bicycle sharing requires a high quality of service and low operational costs by making sure that citizens can always pick up a bicycle in a station and can always return it back to another one without the station exceeding the capacity of parked bicycles. In other words,

[8]Available upon request at http://www.pnwsmartgrid.org/participants.asp (last accessed: March 2017)

[9]For example, the plan shifted 30 minutes into the future is the measured demand for the duration 01:00-12:00.

[10]The algorithm terminates after 26 ± 3.6 and 38 ± 4.1 iterations according to the second termination criterion of Section III-C3.

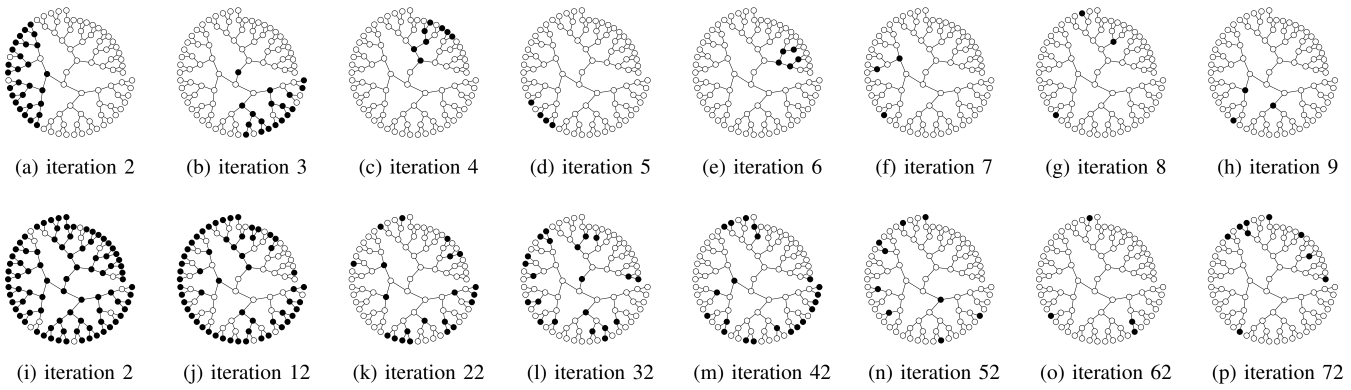


Fig. 11: (a)-(h) I-EPOS vs. (i)-(p) COHDA. Network snapshots showing the agents (in black) changing their selection and agents (in white) that do not change their selection.

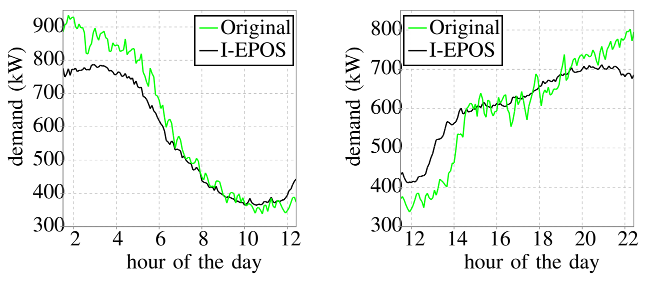


Fig. 12: Power peak-shaving by I-EPOS on the PNW dataset.

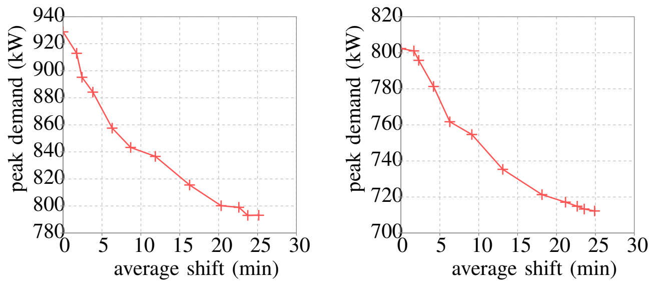


Fig. 13: Trade-off between peak demand reduction and average demand shift in the selected plans $\lambda \in \{0, 0.5, \dots, 4, 5, 6, 100\}$ after $t = 60$ iterations for the PNW datasets.

station should remain load-balanced under various conditions, such as population density, mobility, weather etc. Manual relocation of bicycles by system operators in not viable in the long term and can increase operational costs significantly.

In the context of bicycle sharing, the possible plans may concern user recommendations about the stations from which bicycles are picked up and to which they are returned. The possible plans are encoded as a vector with values the incoming minus the outgoing bicycles of a user in each station at a certain time slot. For example, a user 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 user agent[11] such that the number of bicycles among the stations remains balanced. This can be formalized as minimizing the variance of the global response. The plan dimension here is the stations in contrast to the energy domain in which load-balancing over time is performed.

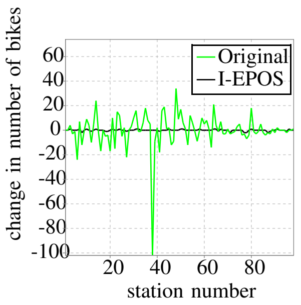
I-EPOS generates bicycle sharing plans by reasoning based on real-world historical data[12] from the Hubway bicycle sharing system in Paris. Although this dataset does not contain personalized records, user trips are extracted from user information: zip-code, year of birth and gender. All trips that have common values in these fields are assumed to be made by the same user. A random subset of 1000 unique users is used as 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 a user 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 users. The local cost of each plan is defined by the likelihood the user does not make the trip instructed in the plan. For instance, if three plans are chosen in 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 14a illustrates the load-balancing of the stations using I-EPOS without varying the local cost of the plans 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 user is unlikely to choose a certain trip, i.e. a trip with high local cost. Figure 14b shows the trade-off between global and local cost controlled via the λ parameter after 30 iterations. A $\lambda = 0$ is the one extreme in which the local cost is not considered, in contrast to $\lambda = 100$

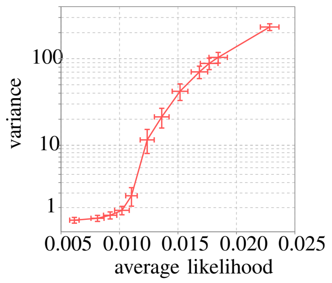
[11]Such an agent can be implemented as a mobile app, for instance.

[12]The dataset is made available in the context of the Hubway Data Visualization Challenge: http://hubwaydatachallenge.org/ (last accessed: March 2017).

that results in global response equivalent to the original data.



(a) Original bicycle allocation 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. 14: I-EPOS performance for the bicycle sharing dataset.

Results show that making plan selection 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.

V. I-EPOS as a Paradigmatic Artifact

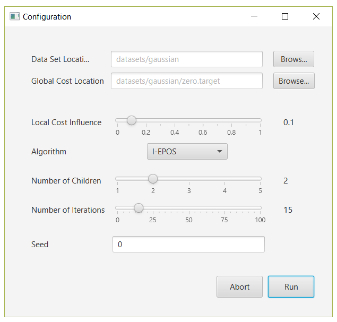
Section IV-B confirms that solutions to decentralized combinatorial optimization problems have a tremendous potential to build more sustainable and resilient digital societies [27]. Authors here move a step forward to contribute a paradigmatic software implementation of I-EPOS together with other supporting 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[13] is a generic and modular open source Java implementation[14] 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[15] and high-quality videos for a visual comprehension of the self-adaptive learning process. The software suite comes with a simulation as well as an actual prototype of I-EPOS designed with a distributed prototyping toolkit [28] for deployment in real-world testbeds such as Planetlab[16]. The community can also make use of integrated plotting and graph visualization capabilities as in Figure 11. Finally, the software suite comes with a graphical user interface for interactive executions as shown in Figure 15.

[13]Available at http://epos-net.org/shared/I-EPOS.zip (last accessed: March 2017).

[14]Available at https://github.com/epournaras/EPOS (last accessed: March 2017).

[15]Available at https://github.com/epournaras/EPOS-Manual (last accessed: March 2017)

[16]Available at https://www.planet-lab.org (last accessed: March 2017). 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: March 2017).



Data Set Locati... datasets/gaussian

Global Cost Location datasets/gaussian/zero.target

Local Cost Influence 0.1

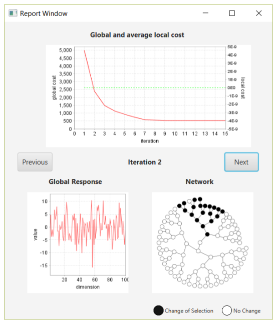
Algorithm I-EPOS

Number of Children 2

Number of Iterations 15

Seed 0

(a) Input



(b) Output

Fig. 15: The graphical user interface of I-EPOS.

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 neighbor 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 cost function `func.VarCostFunction` or the `func.SqrDistCostFunction` that minimizes the squared Euclidean distance from a target incentive signal [12], [15]. 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 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 prototyping of a real-world distributed implementation of I-EPOS significantly improves the technical readiness level of a future community contribution.

VI. Conclusion and Future Work

This paper concludes that the fully decentralized self-adaptive learning process in the challenging combinatorial optimization of I-EPOS is feasible and can even significantly outperform other related algorithms that either make use of non-local brute-force operations or exchange full information. In contrast to discriminatory big data profiling or deep learning AI systems that often overtake control and undermine autonomy, I-EPOS promotes participation, self-determination and autonomy, while it is highly scalable and relies on the exchange of entirely local and aggregated information. These system properties allow new novel disruptive designs for participatory sharing economies in the context of smart grids and smart cities such as energy self-management or self-regulation of urban qualities via bicycle sharing. 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 decentralized learning and combinatorial optimization.

Future work includes the evaluation of new global/local cost functions, the performance comparison of building and maintenance mechanisms for tree topologies in different network settings and the exploration of other application domains. The interplay of the scientific aspects of I-EPOS with art can provide new means for the general public to conceive a self-adaptive learning process that is 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 [29].

References

[1] E. Pournaras and J. Espejo-Uribe, "Self-repairable smart grids via online coordination of smart transformers," *IEEE Transactions on Industrial Informatics*, 2016.

[2] E. Pournaras, B.-E. Brandt, M. Thapa, D. Acharya, J. Espejo-Uribe, M. Ballandies, and D. Helbing, "Sfina-simulation framework for intelligent network adaptations," *Simulation Modelling Practice and Theory*, vol. 72, pp. 34–50, 2017.

[3] C. M. de Chardon, G. Caruso, and I. Thomas, "Bike-share rebalancing strategies, patterns, and purpose," *Journal of Transport Geography*, vol. 55, pp. 22–39, 2016.

[4] J. Puchinger and G. R. Raidl, "Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*. Springer, 2005, pp. 41–53.

[5] W. Yeoh, A. Felner, and S. Koenig, "BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 591–598.

[6] A. Chechetka and K. Sycara, "No-commitment branch and bound search for distributed constraint optimization," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, 2006, pp. 1427–1429.

[7] A. Petcu and B. Faltings, "A scalable method for multiagent constraint optimization," *Artificial Intelligence*, pp. 266–271, 2005.

[8] E. Pournaras, M. Warnier, and F. M. Brazier, "Local agent-based self-stabilisation in global resource utilisation," *International Journal of Autonomic Computing*, vol. 1, no. 4, pp. 350–373, 2010.

[9] E. Pournaras, "Multi-level reconfigurable self-organization in overlay services," Ph.D. dissertation, TU Delft, Delft University of Technology, 2013.

[10] C. Hinrichs, S. Lehnhoff, and M. Sonnenschein, "COHDA: A combinatorial optimization heuristic for distributed agents," in *International Conference on Agents and Artificial Intelligence*. Springer, 2013, pp. 23–39.

[11] ——, "A decentralized heuristic for multiple-choice combinatorial optimization problems," in *Operations Research Proceedings 2012*. Springer, 2014, pp. 297–302.

[12] E. Pournaras, M. Vasirani, R. E. Kooij, and K. Aberer, "Decentralized planning of energy demand for the management of robustness and discomfort," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2280–2289, 2014.

[13] A. Pandey, G. A. Moreno, J. Cámara, and D. Garlan, "Hybrid planning for decision making in self-adaptive systems," in *10th International Conference on Self-adaptive and Self-organizing Systems*. IEEE, 2016.

[14] J. Cámara, D. Garlan, B. Schmerl, and A. Pandey, "Optimal planning for architecture-based self-adaptation via model checking of stochastic games," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, ser. SAC '15. New York, NY, USA: ACM, 2015, pp. 428–435.

[15] E. Pournaras, M. Vasirani, R. E. Kooij, and K. Aberer, "Measuring and controlling unfairness in decentralized planning of energy demand," in *Energy Conference (ENERGYCON), 2014 IEEE International*. IEEE, 2014, pp. 1255–1262.

[16] E. Pournaras, M. Warnier, and F. M. Brazier, "Adaptive self-organization in distributed tree topologies," *International Journal of Distributed Systems and Technologies (IJDST)*, vol. 5, no. 3, pp. 24–57, 2014.

[17] ——, "Adaptation strategies for self-management of tree overlay networks," in *2010 11th IEEE/ACM International Conference on Grid Computing*. IEEE, 2010, pp. 401–409.

[18] E. J. Chang, "Echo algorithms: depth parallel operations on general graphs," *IEEE Transactions on Software Engineering*, vol. 8, no. 4, p. 391, 1982.

[19] Y.-M. Li, Y. Tan, and P. De, "Self-organized formation and evolution of peer-to-peer networks," *INFORMS Journal on Computing*, vol. 25, no. 3, pp. 502–516, 2013.

[20] O. Scekic, H.-L. Truong, and S. Dustdar, "Incentives and rewarding in social computing," *Communications of the ACM*, vol. 56, no. 6, pp. 72–82, 2013.

[21] A. E. Bryson and Y.-C. Ho, *Applied optimal control: optimization, estimation and control*. Xerox College Publishing, 1969.

[22] F. Ducatelle, G. Di Caro, and L. M. Gambardella, "Using ant agents to combine reactive and proactive strategies for routing in mobile ad-hoc networks," *International Journal of Computational Intelligence and Applications*, vol. 5, no. 02, pp. 169–184, 2005.

[23] R. GhasemAghaei, M. A. Rahman, W. Gueaieb, and A. El Saddik, "Ant colony-based reinforcement learning algorithm for routing in wireless sensor networks," in *2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007*. IEEE, 2007, pp. 1–6.

[24] I. Dusparic and V. Cahill, "Distributed w-learning: Multi-policy optimization in self-organizing systems," in *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 2009, pp. 20–29.

[25] H. Hu, J. Zhang, X. Zheng, Y. Yang, and P. Wu, "Self-configuration and self-optimization for lte networks," *IEEE Communications Magazine*, vol. 48, no. 2, pp. 94–100, 2010.

[26] A. Kailas, V. Cecchi, and A. Mukherjee, "A survey of communications and networking technologies for energy management in buildings and home automation," *Journal of Computer Networks and Communications*, vol. 2012, 2012.

[27] D. Helbing and E. Pournaras, "Society: Build digital democracy," *Nature*, vol. 527, pp. 33–34, 2015.

[28] W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer, "Protopeer: a p2p toolkit bridging the gap between simulation and live deployement," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 60.

[29] M. Koutsomichalis and E. Pournaras, "The sound of decentralization-sonifying computational intelligence in sharing economies," in *Proceedings of the 23rd International Symposium on Electronic Art (ISEA 2017)*, 2017.