# Near-Optimal Decentralized Power Supply Restoration in Smart Grids

Pritee Agrawal, Akshat Kumar, Pradeep Varakantham
School of Information Systems, Singapore Management University, Singapore
priteea.2013@phdis.smu.edu.sg,{akshatkumar, pradeepv}@smu.edu.sg

## ABSTRACT

Next generation of smart grids face a number of challenges including co-generation from intermittent renewable power sources, a shift away from monolithic control due to increased market deregulation, and robust operation in the face of disasters. Such heterogeneous nature and high operational readiness requirement of smart grids necessitates decentralized control for critical tasks such as power supply restoration (PSR) after line failures. We present a novel multiagent system based approach for PSR using Lagrangian dual decomposition. Our approach works on general graphs, provides provable *quality-bounds* and requires only *local* message-passing among different connected sub-regions of a smart grid, enabling decentralized control. Using these quality bounds, we show that our approach can provide *near-optimal* solutions on a number of large real-world and synthetic benchmarks. Our approach compares favorably both in solution quality and scalability with previous best multiagent PSR approach.

## Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems, Intelligent agents

## General Terms

Algorithms, Theory

## Keywords

Multi-agent Systems, Distributed Constraint Optimization, Power restoration, Smart Grids

## 1. INTRODUCTION

Due to increased world-wide incentive for cleaner electricity generation [13, 16, 8], next generation of smart grids would feature co-generation from intermittent renewable power sources. In addition, the deregulation of power markets has enabled the presence of multiple operators [4, 9], which marks a shift away from highly regulated monopolies of power grids. Such heterogeneous structure of future smart grids where multiple operators control different sub-regions of the grid presents a unique opportunity for agent-based *decentralized control* of smart grids. Decentralized control of power grids entails high operational readiness, increased robustness and faster response time after disasters. In fact, the importance of such decentralised control has been already recognized in power systems and multiagent systems community [14, 9, 13, 12].

In our work, we address the important problem of power supply restoration (PSR), where a power grid has to be reconfigured after multiple line failures subject to constraints such as acyclic power flow and line capacities [18, 1, 17]. In particular, we focus on the *multiagent* PSR problem where different sub-regions of the grid are controlled by different agents [9]. The power supply needs to be restored to de-energized areas by local coordination among agents without the oversight of a central authority. Such multiagent PSR problems have been previously addressed using the framework of distributed constraint optimization [9, 12]. However, the scalability of such previous approaches is limited due to the NP-Hard [5] complexity of solving PSR problems.

We use the framework of *Lagrangian relaxation* or dual decomposition [2], to solve the multiagent PSR problem. Lagrangian relaxation (LR) has a rich history in power networks community [6, 7, 15]. These previous approaches use the LR approach to solve the problem of optimal power flow (OPF), also known as the optimal dispatch problem. Our application of LR to the PSR problem is significantly different than OPF, as power restoration is a *discrete optimization* problem where we are changing the underlying structure of active power lines in the network. The OPF problem is concerned with deciding how much power each generation unit in the grid must generate to ensure that all the demand is satisfied [15]. Thus, OPF problem is a continuous, albeit, non-convex problem. Such *discrete* versus *continuous* nature between PSR and OPF also gets magnified when we develop strategies to extract a feasible PSR solution during each iteration of LR.

Our main contribution in this paper is to develop a decentralised Lagrangian Relaxation mechanism to solve the PSR problem. A key component that enables decentralisation of LR is the extraction of optimal feasible PSR solution at each iteration of LR in a decentralised manner. Extracting optimal feasible PSR solution even for a tree-structured network is NP-Hard. Fortunately, it is an *easy* NP-Hard problem and we develop a decentralized fully polynomial time approximation scheme (FPTAS) to find a near-optimal feasible PSR solution by exploiting its connections with the knapsack problem. Our decentralised LR mechanism has multiple necessary and important properties, especially in the context of solving the multi agent PSR problem:

- We only use *local* message-passing among agents (each representing a different region) to extract near-optimal feasible PSR solution, thereby enabling decentralised control.
- We are able to provide provable guarantees on solution quality.
- Due to iterative nature of LR, our approach has the desirable

**Appears in:** *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

*anytime* property.

- Our approach is significantly more scalable than previous multi-agent approaches and can solve existing real-world benchmarks near optimally with significant speedups and with significantly low message-passing overhead.

Our decentralised LR approach for the multi agent PSR problem bears some similarity to distributed constraint optimization (DCOP) approaches that partially centralize the optimization problem, such as the asynchronous partial overlay (APO) algorithm [10, 11]. A key difference in our work is that the sub-regions of a power network remain fixed with the LR approach not requiring any additional centralization, whereas the APO approach dynamically changes the (partial) centralization while solving the underlying DCOP problem. Furthermore, as highlighted in [9], a straightforward conversion of the PSR problem to a DCOP presents several challenges. For example, the resulting DCOP has high arity constraints to represent the flow conservation and the line capacity requirement of the PSR problem. Furthermore, discretization of the line capacity is also required to allow standard DCOP approaches to solve PSR problems. In contrast, the LR technique we develop does not require any discretization, and results in a simple distributed approach that works by passing messages among different connected sub-regions of the smart grid.

We test our decentralised LR approach on real world and large synthetic benchmarks [5, 9] and were able to show that our approach is significantly more scalable than previous multi-agent approaches while obtaining near optimal solutions. To show that our approach is competitive with centralized solvers, we also compare it against a highly efficient centralized math programming solver CPLEX. We show that our approach can achieve near optimal solutions, close to 90% optimality, within comparable runtimes as the CPLEX.

## 2. MULTIAGENT PSR PROBLEM

We first describe the centralised PSR problem and then describe the multi-region decomposition that results in multi agent PSR.

A power distribution system is a network of electric lines connected by switching devices (SDs), and fed by circuit breakers (CBs) [1]. Both SDs and CBs have two device positions: closed, open. SDs are analogous to sinks (transformer stations) which consume some power and forward the rest on other lines if closed. Open SD stops power flow. Circuit breakers, which are analogous to power sources with finite available power, feed the network when closed. The positions of the devices are set such that the paths taken by the power of each CB forms a tree called *feeder tree*, and no sink is powered by more than one power line. In addition, Kirchhoff's law (or flow conservation) must hold for all devices and the current load for any line must not exceed its capacity. The problem of power supply restoration (PSR) is that of reconfiguring the network (setting positions of devices) such that power supply is restored to maximum possible affected sinks after one or more lines become faulty. Note that in this work, we assume that faulty elements are identified in the network and isolated to prevent feeding power to a faulty line before the PSR task.

We view a power distribution network as a graph $G = (V = \mathcal{P} \cup \mathcal{S}, E)$. Vertices represent power sources (CBs) $p_i \in \mathcal{P}$ and sinks (SDs) $s_i \in \mathcal{S}$. Each power source has a finite amount $|p_i|$ of power available. A sink $s_i$ consumes $|s_i|$ units of power. A value $v_i$, called *sink weight*, is also associated with a sink $s_i$ to denote the relative importance of the sink. Edges represent power lines connecting sinks and power sources. Let $L$ denote the power capacity of a line. For ease of exposition, we assume it is the same for each line. There is a positive $\epsilon > 0$ line loss associated with power flow across a line.

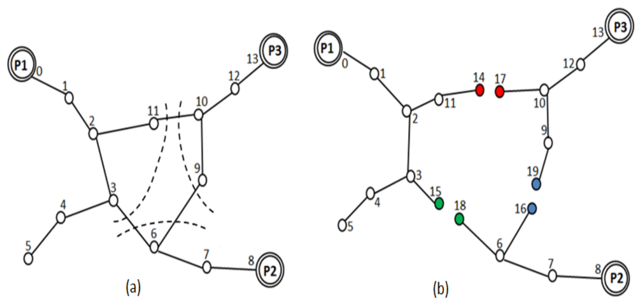


**Figure 1: Multi-region decomposition of a power network using colored *relay nodes* on the right**

We now describe the multiagent PSR problem. Due to the presence of underlying decomposition (due to de-regulation), we have a region-based decomposition of the underlying graph $G$. The vertices of the network $G$ are partitioned into $\mathcal{R}$ regions: $V = \cup_{r\in\mathcal{R}} V^r$. Let $E^r$ denote the edge set such that both its vertices lie in the region $r$. Intuitively, each region $r$ is managed by a different entity. Therefore, each region $r$ represents an *agent* in our multiagent PSR problem. The edges in set $E \setminus \cup_{r\in\mathcal{R}} E^r$ denote the *cut edges* that connect different regions. Power can flow from one region to another region via these cut edges. Analogously, only agents that share cut edges can communicate with each other along such edges, resulting in a *multiagent system* representation of a power network.

**Relay Nodes:** Our approach to solve the multiagent PSR via message-passing along the cut edges is to view each network region as a *separate* sub-network per agent. Intuitively, cut edges are the *complicating* edges that connect two different regions. It is not clear whose agent's sub-network they belong to. To make separation among regions clear and exploitable by optimization algorithms later, we introduce the notion of *relay nodes*. A pair of relay nodes are defined for each cut edge $(u^r, v^{r'})$ connecting regions $r$ and $r'$. One relay node $c^r$ belongs to the region $r$ and the second relay node $c^{r'}$ belongs to region $r'$. We then remove the cut edge $(u^r, v^{r'})$ from the graph $G$, and create two new *relay* edges $(u^r, c^r)$ and $(c^{r'}, v^{r'})$. Figure 1 describes such a process of decomposing the original network $G$ into multiple independent networks $G^r$, one for each region $r \in \mathcal{R}$. Relay nodes can consume any amount of power that comes in, and also act as power source with infinite supply. Even though we have partitioned the original network $G$ into independent regions, flow conservation constraints for relay nodes still logically connect the whole network together. Intuitively, the total incoming power for a relay node $c^r$ in region $r$ must be equal to the total outgoing power for its paired relay node $c^{r'}$ in region $r'$.

In the next section, we provide a mixed-integer linear program (MILP) to solve PSR for a network decomposed into multiple regions using relay nodes. Naturally, this approach is not a decentralized approach. However, using the technique of Lagrangian relaxation (LR) [2], we relax the flow conservation constraints for relay nodes. We then show that the resulting dual problem can solved *independently* for each region $r$ via local message-passing along the cut edges. Thus, introducing the idea of relay nodes in a global network divided into multiple regions leads to a decentralized algorithm within the LR framework.

## 3. DUAL DECOMPOSITION FOR MULTIAGENT PSR

| Variable | Definition |
|---|---|
| $x_i^r$ | $x_i^r = 1$ indicates that sink $s_i$ in region $r$ consumes $\lvert s_i\rvert$ units of power |
| $x_{ij}^r$ | $x_{ij}^r$ = 1 indicates that power flows from node $i$ to $j$ in region $r$ |
| $d_{ij}^r$ | Power flow between nodes $i$ and $j$ in region $r$ |
| $\mathcal{S}^r$ | Set of sinks in region $r$ |
| $\mathcal{P}^r$ | Set of power sources in region $r$ |
| $\mathcal{C}^r$ | Set of relay nodes in region $r$ |
| $\mathcal{E}^r$ | Set of relay edges in region $r$ |
| $\delta_{ij}^r$ | Intermediate variable employed to linearize $x_{ij}^r \cdot d_{ij}^r$ |

**Table 1: Notation**

We first develop a MILP to solve the PSR problem for the decomposed global network. We then show how to relax the complicating constraints within this MILP such that the resulting dual problem can be solved in a decentralized manner. Before we introduce the MILP, we provide the notation employed in Table 1. Let $\mathcal{C}^r$ denote the set of all relay nodes and $\mathcal{E}^r$ denote the set of all relay edges belonging to a region $r$. Each region $r$ is then described using the graph $G^r = (V^r \cup \mathcal{C}^r, E^r \cup \mathcal{E}^r)$. The node set $V^r$ consists of sink set $\mathcal{S}^r$ and power source set $\mathcal{P}^r$ for the region $r$.

Table 2 shows the math program for finding the optimal solution to the PSR problem. The objective of this program is to maximize the total weight of nodes that are supplied power. Binary decision variables $x_i$ are created for each sink and relay node for each region (superscripts denote region). If $x_i = 1$, then sink $s_i$ consumes $|s_i|$ units of power. Binary variables $x_{ij}$ are created for each edge in the network for a region $r$. If $x_{ij} = 1$, it means that power flows from node $i$ to $j$. The continuous variable $d_{ij}$ denotes the amount of power flow along the directed edge $(i, j)$. The constraints in table 2 denote the following:

- Constraint 2 denotes that a sink is always switched off if there is no incoming power flow, otherwise, it can be switched on.
- Constraint 3 denotes that a sink can receive power from at most one of its neighbors. This is also a necessary constraint for PSR. This also helps maintain the acyclicity of power flow.
- Constraint 4 is the capacity constraint for a line.
- Constraint 5 denotes flow conservation for sink nodes. For every sink $s_j$, total power inflow must be equal to the sum of power consumed by it, the line loss $\epsilon$ for incoming power, and the total power forwarded to other nodes.
- Constraint 6 denotes flow conservation for relay nodes. As relay nodes are created in *pairs*, we use the terminology that for every relay node $c \in \mathcal{C}^r$ in region $r$, its pair is denoted as $c' \in \mathcal{C}^{r'}$ in region $r'$[1]This flow constraint denotes that the total *incoming* power into relay node $c$ in region $r$ should be equal to the total outgoing power from $c'$ in region $r'$.
- Constraint 7 is the capacity constraint for all the power sources across all regions $p_i \in \mathcal{P}$.The total outgoing power from a power source should not exceed the power available to it.

Notice that the math program in table 2 is nonlinear due to quadratic terms, such as $x_{ij}^r \cdot d_{ij}^r$ in constraints. However, this constraint can be linearized as the variables $x_{ij}^r$ are binary and the power flow $d_{ij}^r$ is bounded. To linearize such quadratic terms, we replace $x_{ij}^r \cdot d_{ij}^r$ by a new variable $\delta_{ij}^r$. The constraints (5), (6) and (7) in the previous formulation are thus replaced with following constraints (9), (10) and (11). To complete the linearization, we also add a new constraint (12). The constant $M$ is a large number. Such constraints are also known as 'big M' constraints in the OR literature.

Variables: $x_i^r\ \forall s_i^r \in \mathcal{S}^r \cup \mathcal{C}^r;\ x_{ij}^r, d_{ij}^r\ \forall (i,j) \in E^r \cup \mathcal{E}^r\ \forall r \in \mathcal{R}$

Minimize:
$$-\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{S}^r} x_i^r \cdot v_i^r \tag{1}$$

Subject to:
$$x_j^r \le \sum_{i\in V^r\cup\mathcal{C}^r} x_{ij}^r \quad ,\forall j \in \mathcal{S}^r\cup\mathcal{C}^r, \forall r \tag{2}$$
$$\sum_{i\in V^r\cup\mathcal{C}^r} x_{ij}^r \le 1 \quad ,\forall j \in \mathcal{S}^r\cup\mathcal{C}^r, \forall r \tag{3}$$
$$d_{ij}^r \le L \quad \forall (i,j)\in E^r\cup\mathcal{E}^r, \forall r \tag{4}$$
$$\sum_{i\in V^r\cup\mathcal{C}^r}(d_{ij}^r-\epsilon)\cdot x_{ij}^r = \sum_{l\in\mathcal{S}^r\cup\mathcal{C}^r} d_{jl}^r \cdot x_{jl}^r + x_j^r \cdot |s_j^r|, \forall j \in \mathcal{S}^r, \forall r \tag{5}$$
$$\sum_{i\in V^r}(d_{ic}^r-\epsilon)\cdot x_{ic}^r = \sum_{j\in\mathcal{S}^{r'}\cup\mathcal{C}^{r'}} d_{c'j}^{r'} \cdot x_{c'j}^{r'}, \forall c \in \mathcal{C}^r, \forall r \tag{6}$$
$$\sum_{j\in\mathcal{S}^r\cup\mathcal{C}^r} d_{pj}^r \cdot x_{pj}^r \le |p| \ ,\forall p \in \mathcal{P}^r, \forall r \tag{7}$$
$$d_{ij}^r \ge 0 \quad ,x_{ij}^r \in \{0,1\} \quad ,x_i^r \in \{0,1\} \tag{8}$$

**Table 2: Nonlinear mathematical program for PSR**

$$\sum_{i\in V^r\cup\mathcal{C}^r}\delta_{ij}^r - \epsilon\cdot x_{ij}^r = \sum_{l\in\mathcal{S}^r\cup\mathcal{C}^r}\delta_{jl}^r + x_j^r\cdot|s_j^r|, \forall j\in\mathcal{S}^r, \forall r \tag{9}$$
$$\sum_{i\in V^r}\delta_{ic}^r - \epsilon\cdot x_{ic}^r = \sum_{j\in\mathcal{S}^{r'}\cup\mathcal{C}^{r'}}\delta_{c'j}^{r'}, \forall c\in\mathcal{C}^r, \forall r \tag{10}$$
$$\sum_{j\in\mathcal{S}^r\cup\mathcal{C}^r}\delta_{pj}^r \le |p| \ ,\forall p\in\mathcal{P}^r, \forall r \tag{11}$$
$$\delta_{ij}^r \begin{cases} \le d_{ij}^r, & \forall (i,j)\in E^r\cup\mathcal{E}^r, \forall r \\ \le M\cdot x_{ij}^r & \forall (i,j)\in E^r\cup\mathcal{E}^r, \forall r \\ \ge d_{ij}^r + (x_{ij}^r - 1)\cdot M, & \forall (i,j)\in E^r\cup\mathcal{E}^r, \forall r \end{cases} \tag{12}$$

We refer to the program of table 2 with the above linearized constraints as the *global MILP* for PSR.

**Model Extensions** The basic model we presented in table 2 can be extended in multiple ways to account for varying preferences of power grid operators. For example, one can include linear terms in the objective function that penalize switching of devices from their previous positions to minimize the number of device switching operations in the network. We used a simple approximation of the line loss. In networks, where accurate modeling of line loss is required, a linear approximation of the line loss can be added to flow equations [3, 17].

### 3.1 Relaxing Flow Conservation For Relay Nodes

Notice that every constraint and variable in the global MIP is separable for each region $r$, except for the flow conservation constraint for relay nodes (10). Therefore, if we are able to principally *relax* this constraint, then our global MIP would decompose into independent parts, one per region, which can be solved independently of each other. This is indeed achievable by using the technique of Lagrangian relaxation (LR) or dual decomposition [2].

[1]For ease of exposition, for every region $r$, we assume that there is at most one connecting region $r'$. In general, there can be multiple regions $r'$ that share a cut edge with $r$ and we can use multiple flow conservation constraints (6) to represent such connectivity.

We first provide a brief overview of the LR approach, more details can be found in [2, Chapter 6]. Consider the optimization problem:

$$\min f(x)$$
$$\text{s.t} \quad x \in X, \;\; g_j(x) \leq 0, \; j = 1, \ldots, p$$

The dual function $q(\cdot)$ and the Lagrangian $L(\cdot)$ of the above problem after dualizing all the constraints $g$ are given as:

$$q(\mu) = \inf_{x \in X} L(x, \mu) = \inf_{x \in X} \big\{ f(x) + \mu \cdot g(x) \big\} \tag{13}$$

The dual solution $q(\mu)$ is a lower bound of optimal $f^\star(x)$ for every value of dual variables $\mu$. The advantage while working with the dual formulation of the original problem is that the structure of the dual is often much simpler leading to computational gains. Furthermore, the dual solution also provides a lower bound on the original problem. In addition, the dual optimization problem, $\max_{\mu: q(\mu) > -\infty} q(\mu)$, is always concave, and can be solved optimally using the projected subgradient method even in the case of non-differentiable objective function [2].

We therefore relax or *dualize* the complicating flow conservation constraint (10) for all the relay nodes in each region. For each complicating constraint, we create a dual variable $\lambda_c^r \; \forall c \in \mathcal{C}^r, \; \forall r$. This dual variable can be thought of as the cost of violating the flow conservation constraint. The dual decomposition technique will try to find the 'right' cost such that violations of the dualized constraints are minimal. The Lagrangian function, $L(\{\boldsymbol{x}^r, \boldsymbol{\delta}^r, \boldsymbol{d}^r\}, \boldsymbol{\lambda})$, (ignoring the line loss $\epsilon$ for ease of exposition) is given as:

$$\sum_{r \in \mathcal{R}, i \in \mathcal{S}^r} -x_i^r v_i^r + \sum_{r \in \mathcal{R}, c \in \mathcal{C}^r} \lambda_c^r \Big( \sum_j \delta_{c'j}^{r'} - \sum_i \delta_{ic}^r \Big) \tag{14}$$

Upon rearranging the terms to highlight the separable structure of Lagrangian further, we simplify the above to get:

$$\sum_{r \in \mathcal{R}} \Bigg\{ \sum_{i \in \mathcal{S}^r} -x_i^r v_i^r + \sum_{c \in \mathcal{C}^r} \Bigg( \sum_j \delta_{cj}^r \cdot \lambda_{c'}^{r'} - \sum_i \delta_{ic}^r \cdot \lambda_c^r \Bigg) \Bigg\} \tag{15}$$

Notice that in the above equation, nodes $c$ and $c'$ are paired relay nodes in region $r$ and $r'$. Using the above equation, we get our dual $q(\boldsymbol{\lambda})$ as below:

$$\sum_{r \in \mathcal{R}} \min_{\boldsymbol{x}^r, \boldsymbol{\delta}^r, \boldsymbol{d}^r} \Bigg\{ \sum_{i \in \mathcal{S}^r} -x_i^r v_i^r + \sum_{c \in \mathcal{C}^r} \Bigg( \sum_j \delta_{cj}^r \cdot \lambda_{c'}^{r'} - \sum_i \delta_{ic}^r \cdot \lambda_c^r \Bigg) \Bigg\} \tag{16}$$

Notice that the above dual function can be evaluated by solving the inner optimization problem *independently* for each region $r$ controlled by the corresponding agent. Thus, evaluating the dual is substantially simplified, and follows the region-based decomposition of the global network $G$.

**Practical Considerations** The performance of the LR approach w.r.t. the solution quality is affected by the number of constraints that are being relaxed or dualized. In general, the optimal dual solution $q^\star$ may not be equal to the optimal primal solution $p^\star$. The gap between these solutions $(p^\star - q^\star)$, also called the duality gap, can increase with higher number of relaxed constraints. Nonetheless, we show empirically that for several large instances, the LR approach is able to provide good solution quality despite large number of dualized constraints. The degradation in the performance of LR with increasing number of relaxed constraints is graceful.

## 3.2 Maximizing the Dual Function

As the dual function is a lower bound on the optimal primal optimal solution for every value of $\boldsymbol{\lambda}$, we now address the problem of optimizing dual: $\max_{\boldsymbol{\lambda}} q(\boldsymbol{\lambda})$. We maximize the dual iteratively by using the projected sub-gradient ascent technique [2]. Using the sub-gradient information, the updated value of the dual variable $\lambda_c^{r\star}$ for the next iteration is:

$$\lambda_c^{r\star} = \lambda_c^r + \alpha \Bigg[ \sum_{j \in \mathcal{S}^{r'} \cup \mathcal{C}^{r'}} \bar{\delta}_{c'j}^{r'} - \sum_{i \in V^r} \bar{\delta}_{ic}^r \Bigg] \; \forall c \in \mathcal{C}^r, \forall r \tag{17}$$

where $\bar{\delta}_{c'j}^{r'}$ and $\bar{\delta}_{ic}^r$ are the variable values obtained while solving the corresponding minimization problem in (16) for regions $r'$ and $r$ respectively; $c$ and $c'$ are paired relay nodes in regions $r$ and $r'$. The parameter $\alpha$ is the step parameter. Notice that such an update of dual variables requires exchange of variable values only across the cut edges of the global network among neighboring agents. Thus, this step can be carried out in a distributed fashion.

The step size $\alpha$ is crucial to fast and accurate convergence of the LR approach. There are a number of recommendations for setting the step size in sub-gradient method. We use the following rule which has theoretical justifications in [2]. The step size $\alpha^{i+1}$ for the iteration $i+1$ is set based on quantities computed in iteration $i$ as follows:

$$\alpha^{i+1} = \frac{Primal^i - Dual^i}{||\nabla q^i||^2} \tag{18}$$

where $Primal^i$ denotes the primal solution quality (or the solution quality corresponding to a feasible solution satisfying all the power network constraints) and $Dual^i$ denotes the dual value $q(\boldsymbol{\lambda}^i)$ for the current iteration $i$, and $\nabla q^i$ denotes the sub-gradient of the dual function $q$. The sub-gradient w.r.t. a variable $\lambda_c^r$ is essentially the quantity in square brackets in Eq. (17). Each quantity in Eq. (18) is readily available except the primal solution corresponding to the current iteration's dual solution. Therefore, we need to extract a good primal solution at each iteration, which also the imparts the desirable anytime property to the LR approach. Thus, at each iteration, our approach provides a feasible reconfiguration plan corresponding to the primal solution and the quality bounds using the dual solution $q(\boldsymbol{\lambda})$.

## 3.3 Extraction of Feasible Primal Solution

At each iteration of our LR approach, we need to extract a good quality feasible primal solution that satisfies all power network constraints from the current dual solution. It is challenging to extract a feasible solution from the dual solution as flow conservation will be violated for relay nodes. Our strategy is to first extract one feeder tree for each power source in the global network. This can be done in a distributed manner by locally inspecting the direction and amount of power flow ($x_{ij}$, $\delta_{ij}$) for relay edges within each region. Once we have such feeder trees extracted from the current dual solution with a power source as root node, we attempt to determine which nodes should be switched on or off such that total sink weight of switched on nodes is maximized. Unfortunately, we show below that even this problem is challenging.

PROPOSITION 1. *Solving optimally the PSR problem is NP-Hard even for tree-structured power networks.*

PROOF. We reduce the well known 0/1 knapsack problem to a tree-structured power network. Consider a generic 0/1 knapsack problem with a set of $m$ items, and two $m$ tuples of positive integers, corresponding to values: $\langle v_1, v_2, \cdots, v_m \rangle$, and weights of items: $\langle w_1, w_2, \cdots, w_m \rangle$. The goal is to identify the set of items whose total weight is less than the given capacity $W$ and it yields the highest value.

For this 0/1 knapsack problem, we have an equivalent PSR problem with graph $G = (\mathcal{P} \cup \mathcal{S}, E)$ where.

- $\mathcal{P} = \{p_1\}$, where $p_1$ has a finite supply $W$.
- $\mathcal{S} = \{s_1, s_2, \cdots, s_m\}$, where $s_i$ consumes $w_i$ units of power and has a sink weight of $v_i$.
- $E = \{(p_1, s_1), (s_1, s_2), (s_2, s_3), \cdots, (s_{m-1}, s_m)\}$ denotes a chain network

We set the line capacity of each edge as $W$. Clearly, the above reduction is a polynomial time. We have the following correspondence between the optimal PSR solution for this chain and the knapsack. If sink $s_i$ consumes power (i.e., $x_i = 1$), then the item $i$ is included in the knapsack, otherwise not. Given that line and power source capacity is $W$, total switched-on sink consumption will always be less than $W$. As we maximize $\sum_i x_i v_i$ for the power network, this implies finding the best set of items to be included in the knapsack. One can show that such an optimal PSR solution is also optimal for knapsack, and vice-versa. Thus, PSR problem is NP-Hard even for tree-structured networks. □

Despite the above negative result, we exploit the fact that the knapsack problem is an *easy* NP-Hard problem and admits a fully polynomial time approximation scheme (FPTAS). Thus, our approach is to derive an FPTAS for optimal PSR in a tree-structured problem. Furthermore, we also show that such a scheme can be implemented using message-passing along the edges of the power network making all the steps of our approach—evaluating the dual, iteratively maximizing the dual and extracting a feasible solution—distributed in nature.

### 3.3.1 Feeder Tree Extraction

We provide a sketch of the procedure to extract feeder trees from the current dual solution. The first step for extracting a feeder tree is to determine the direction of power flow for the cut edges. Notice that for every edge $(i, j)$ for any region that is not a cut edge, the binary variable $x_{ij}$ denotes the power flow direction. However, while decomposing a power network into regions, a cut edge $(u, v)$ that connects two different regions $r$ and $r'$ is being decomposed into two relay edges $(u, c)$ in region $r$ and $(c', v)$ in region $r'$.

We now outline a simple procedure that determines the power flow direction for a cut edge $(u, v)$ based on inspecting the dual solution (the $\delta$ variable) for the relay edges. Figure 2 visually shows this procedure. Based on the dual solution for the relay edges $(u, c)$ and $(c', v)$, there are three cases possible. In the first case, positive power flows from node $u$ to node $c$, and from node $c'$ to node $v$. In this case, we make the direction of flow from the node $u$ to node $v$ for the cut edge. Notice that we are only inspecting the direction of power flow. The amount of power may be inconsistent in the dual solution. For e.g., one relay node may assume that it is forwarding 10 units of power to its paired node, and its paired node may assume that it is receiving 20 units, leading to violation of flow conservation constraints. Correcting such flow conservation violation is exactly the main task for primal extraction.

For the other two cases (case 2 and case 3 in Figure 2), there is no flow across the cut edge and its corresponding devices are set to the open position. Once we have determined the direction of power flow for each edge in the global power network, the only remaining task is to extract a feeder tree corresponding to each power source. This can be done easily as the direction of power flow provides the parent-child relationship required for the tree construction. Furthermore, given the constraint that a node cannot receive power from multiple incoming edges, there are going to be no cycles while extracting such trees. Notice also that we discard any tree and its corresponding nodes where a relay node is the root. For e.g., case 3 in figure 2 denotes one such setting. Once we have such trees extracted, we develop a tree-based dynamic programming algorithm that determines which sinks are switched on and off to maximize the sink weight for each feeder tree.

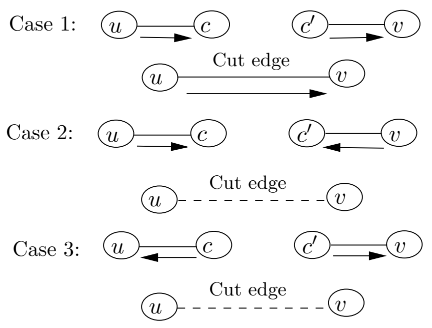


**Figure 2: Determining power flow for the cut edge $(u, v)$ from the dual solution for relay edges $(u, c)$ in region $r$ and $(c', v)$ in paired region $r'$. Arrows denote power flow direction. A dotted line denotes no power flow with corresponding devices being open.**

### 3.3.2 Tree Based Dynamic Programming (TBDP)

Our scheme, called TBDP, provides an FPTAS for a tree-structured PSR. It follows the high level architecture of [19] for stochastic network design in ecology. Our approach is different on multiple fronts due to different recurrence relationships than [19], which are essential for deriving the FPTAS for PSR.

We are given a tree $\mathcal{A} = (V, E)$. The root of this tree is a power source $P$ with maximum supply $p$. We denote other nodes (total of $m$) using $i$ and $j$s. Their sink consumption is $s_i$, value is $v_i$. The capacity of each line is $L$. We denote using $\pi$ the policy that specifies which node should be switched on and thus, consume some power, and forward the rest downstream. If a node is switched off, it does not consume any power and forwards all power to its children. The binary decision variable is $\pi(i)$ indicating whether a sink node is on or off. The *total power* consumed by the entire subtree rooted at (and including) node $i$ be $t(i) = \sum_{j \in \mathcal{T}_i} \pi(j) s_j$. For a given policy $\pi$, the optimization problem to solve is:

$$\max_{\pi} \sum_i \pi(i) v_i \text{ s.t. } t(i) \leq L \;\forall i \in V,\; t(P) \leq p \tag{19}$$

Let $z_i(\pi)$ denote the total utility of the subtree $\mathcal{T}_i$:

$$z_i(\pi) = \sum_{j \in \mathcal{T}_i} \pi(j) v_j$$

We first highlight a simple, yet important recurrence relation for quantities $z(\cdot)$ that forms the basis of the FPTAS for PSR:

$$z_i(\pi) = \pi(i) v_i + \sum_{j \in \text{Ch}(i)} z_j(\pi), \qquad \forall i \tag{20}$$

where $\text{Ch}(i)$ is the set of children of $i$. Another key recurrence relationship is for the *minimum* power supply $C_i(z)$ needed to produce a utility of exactly $z$ from the subtree $\mathcal{T}_i$ rooted at node $i$. Without loss of generality, we assume that each node has at most two children, otherwise, any tree with more than 2 children per node can be reconfigured to a two-children tree by creating dummy nodes as in [19].

$$C_i(z) = \min_{\pi(i) \in \{0,1\}} C_{i,\pi(i)}(z) \tag{21}$$

$$C_{i,\pi(i)}(z) = \min_{0 \le d \le z'_i} \pi(i)s_i + C_j(d) + C_k(z'_i - d) \tag{22}$$

where $z'_i$ is $z - \pi(i)v_i$; $j$ and $k$ are the two children of the node $i$. We can also write a similar recurrence when node $i$ is a leaf node or has only one child (details omitted). The recurrence relationships (20) and (22) for the PSR problem form the basis of key differences in our work from that of [19]. The above recurrences can provide a pseudo-polynomial time dynamic programming approach for optimal PSR similar to the knapsack problem. However, to develop an FPTAS, we need to *discretize* all the possible achievable utilities $z(\pi)$. We write a discretized version of Eq. (20) as:

$$\hat{z}_i(\pi) = K_i \left\lfloor \frac{\pi(i)v_i + \sum_{j \in \mathrm{Ch}(i)} \hat{z}_j(\pi)}{K_i} \right\rfloor \forall i \tag{23}$$

where $K_i$ is a constant provided as input. Using theoretical analysis similar to [19], we can show that $K_i = K = 0.5 \cdot \beta \cdot (\min_{j \in V} v_j)$ provides an FPTAS, where $\beta$ is an input parameter that determines the optimally guarantee for the FPTAS. Once we have discretized the set of all possible utility values using constant $K$, all that is remaining is to perform a bottom-up dynamic programming to solve the problem (21) for each node in the feeder tree, and a final policy determining top-down phase on the feeder tree. These computations can be performed by using message-passing and is again distributed. We omit the details of such bottom-up and top down pass of TBDP as it is similar to [19]. While doing the bottom-up pass, we make sure that if $C_i(z) > L$, then we make it infeasible as $C_i(z) = \infty$ to respect line capacity constraint. Once the $C(z)$ values are computed at the root node, we choose the highest utility $z$ such that $C(z) \le p$, the power capacity of the source. Using similar theoretical analysis provided in [19], our approach can provably provide a solution within $\beta$ fraction of the optimal with a worst-case runtime guarantee of $O(m^2/\beta^2)$ given $m$ nodes in the tree. In practice, TBDP was much faster than the worst-case time.

## 3.4 D$^2$ADP Approach

D$^2$ADP or dual decomposition based approximate dynamic programming uses lagrangian dual decomposition along with dynamic programming in a decentralized manner. The algorithm has four major phases. The first phase involves the SolveRegion() function that computes dual $D$ over all regions by solving each region independently. The second phase involves the FeederTreeExtract() function that extracts the feeder trees $\mathcal{A}$ to determine the parent-child relationship between relay node pairs. We extract one feeder tree for every power source, regardless of the regions. All trees are disjoint (at most one incoming power line for a node) and may contain nodes from different regions (via exchange of messages for relay node pairs). The third phase employs the TBDP() function to compute primal solution $P_{tree}$ for every feeder tree using a bottom up followed by a top down message passing scheme. The bottom up message passing stage finds the best solution where every node sends its reward information to it's parent node. The top down stage of TBDP is responsible for assignment of resources where every power source chooses the best assignment configuration for all nodes that passes down the tree. Finally, the price update phase updates the price variables, $\lambda$ using the price update rule Eq. (17). This process is repeated until convergence which occurs when the duality gap is less than a very small number, $\epsilon$ or when a specified number of iterations is reached.

**Space Complexity Analysis** We provide the space complexity for D$^2$ADP since exchange of messages requires storage space. We show that the space requirement of D$^2$ADP is very small, even in the worst case where the count and size of messages exchanged increase proportionally with the increase in number of cut edges and total nodes respectively.

**Algorithm 1** D$^2$ADP ALGORITHM

```
Initialization : $\boldsymbol{\lambda}^0 \leftarrow 0$
repeat
    $D^r, \boldsymbol{x^r}, \boldsymbol{\delta^r} \leftarrow SolveRegion(\boldsymbol{\lambda^{0,r}}, Region\ r),\ \forall r \in R$
    $\mathcal{A}_{r,p} \leftarrow FeederTreeExtract(\boldsymbol{x^r}, \boldsymbol{\delta^r}),\ \forall p \in \mathcal{P}^r, \forall r$
    $P_{tree} \leftarrow TBDP(\mathcal{A}_{r,p}), \forall p \in \mathcal{P}^r, \forall r$
    $\lambda_c^{r,\star} = \lambda_c^r + \alpha \left[\sum_j \bar{\delta}_{c'j}^{r'} - \sum_i \bar{\delta}_{ic}^r\right], \forall c \in \mathcal{C}^r, \forall r$
until Convergence
return $P, x, \delta$
```

PROPOSITION 2. *Total number of messages exchanged over cut edges is $O(E_c)$, where $E_c$ is total number of cut edges.*

PROOF. All inter-region message exchanges take place only over cut edges. Every relay node obtained from cut edges requires one message exchange for dual variables, one for determining parent-child relationship in feeder tree and 2 for top-down/bottom-up messages over feeder tree. Hence, total messages exchanged is $4E_c$, i.e. $O(E_c)$. $\square$

PROPOSITION 3. *The maximum size of message exchanged over cut edges is $O(N/\beta)$, where $N$ denotes total number of nodes and $\beta$ is the FPTAS parameter.*

PROOF. The maximum size of any message in D$^2$ADP is dominated by dynamic programming for primal extraction which requires top-down and bottom-up message exchanges for feeder trees. Let us consider the worst case where every node is a region and single power source is available with power lines having infinite line capacity. Here, all nodes will be the part of a single feeder tree where the discretized utilities for root node can be obtained as $\{0K_i, 1K_i, ..., l_iK_i\}$ from Eq. (23). $l_iK_i$ provides the upper bound for message size with $l_i$ being an integer value given by $\left\lceil \frac{z_i(\pi)}{K_i} \right\rceil$. We further assume that $m \le v_j \le M$, $\forall j \in \mathcal{T}_i$. Then, $l_i = \left\lceil \frac{z_i(\pi)}{K_i} \right\rceil \le \left\lceil \frac{NM}{K_i} \right\rceil = O\left(\frac{N}{\beta}\right)$ $\square$

## 4. EXPERIMENTS

In this section, we perform experiments[2] to compare D$^2$ADP with the dynamic programming based decentralized approach for power supply restoration in the literature [9], referred to as PSR-DPOP or DPOP for short. We also compare against the centralized global MILP solved using CPLEX. Our goal in these experiments is the following. We demonstrate that our approach, despite being approximate in nature, can provide provably near-optimal solutions for a range of problems. We also show the scalability of our approach on large synthetic benchmarks while providing good quality guarantees, close to 90% optimality. We further highlight the anytime nature of our approach to provide good solutions quickly.

**Real World Benchmarks** We test on real world benchmarks representing real world configurations of NESA, a power distribution company in Denmark [5, 9]. We specifically take the two largest configurations and decompose them into multiple regions. The smaller of the two, 'Large' network, has 56 sinks, 66 lines and 2 power sources. We decompose this network into 7, 10 and 15

[2]All our experiments were performed on a 3.2GHz CPU with 4GB RAM. All our optimization problems are run on CPLEX v12.5

| Configuration | Algorithm | Time(s) | Total(Kb) | Max(Kb) | Optimality |
|---|---|---|---|---|---|
| Large(56R) | DPOP | 54.5 | 36000.0 | 3900.0 | 100% |
| Large(7R) | $D^2$ADP | 8.7 | 119.7 | 0.3 | 97.3% |
| Large(10R) | $D^2$ADP | 6.0 | 149.4 | 0.3 | 92.9% |
| Large(15R) | $D^2$ADP | 8.6 | 178.3 | 0.2 | 89.1% |
| Large(56R) | $D^2$ADP | 17.8 | 639.7 | 0.3 | 87.8% |
| Complex(119R) | DPOP | – | – | – | – |
| Complex(10R) | $D^2$ADP | 15.0 | 219.5 | 0.5 | 91.7% |
| Complex(15R) | $D^2$ADP | 14.8 | 239.3 | 0.4 | 88.4% |
| Complex(20R) | $D^2$ADP | 15.5 | 272.9 | 0.4 | 78.3% |
| Complex(119R) | $D^2$ADP | 41.0 | 1086.3 | 0.5 | 70.1% |

**Table 3: Real World Configurations. The quantities in $(\cdot)$ denote total number of regions for the instance. 'Total' denotes the total size of all the messages exchanged, 'Max' denotes the maximum message size between any two agents.**

underlying regions to test the scalability of our approach with increasing regions. The PSR-DPOP approach of [9] cannot exploit the region based decomposition of the power network, and considers each node in the network to be controlled by a different agent. Therefore, for fair comparisons against PSR-DPOP, we also show results on the 'Large' benchmark with 56 regions (denoted as 'Large(56R)'). The largest instance 'Complex' has 119 sinks, 146 lines and 3 power sources. We consider 10, 15 and 20 regions for this case along with the extreme case in which every network node is a separate agent. We run our approach, $D^2$ADP for 100 iterations. The PSR-DPOP is not an anytime approach, so we run it with the maximum memory limited to 4GB.

Table 3 provides the comparisons against PSR-DPOP on real world configurations, with respect to runtime, space requirements (total and maximum message size), and solution quality. PSR-DPOP was able to terminate and provide a solution only on the 'Large' configuration. As PSR-DPOP is agnostic to region based decomposition of the network, the results for PSR-DPOP are exactly the same for varying number of regions. As PSR-DPOP is an optimal approach, it provided the optimal solution upon termination for 'Large(56R)'. We can clearly see that our approach provides provably near-optimal solutions without the large message overhead of the PSR-DPOP approach. The total message size and the maximum size is significantly smaller for $D^2$ADP.

For the largest 'Complex' instance, PSR-DPOP was unable to provide any solution as its memory requirements are exponential in the tree-width of the underlying network, which was about 40 for the 'Complex' instance. In contrast, our approach scales well for this largest instance with varying number of regions. As highlighted earlier, the performance of the LR approach is adversely affected by the increasing number of regions in the network as this causes several constraints to be relaxed. Nonetheless, $D^2$ADP's performance w.r.t. solution quality varied gracefully while increasing the number of regions from 10 to 119. Notice that the instance 'Complex(119R)' represents the worst possible scenario for our approach as each network node is an agent, thereby relaxing the flow conservation constraint for each edge. Despite this, our approach is able to get a solution provably within 70% of the optimal. We expect that in real networks, such an extreme case is unlikely to occur as each region in a power network is typically composed of multiple nodes.

**Large Synthetic Benchmarks** To further experiment with the scalability, we created synthetic configurations based on the 'Complex' real-world configuration. We refer to them as 'L-Complex(<Number of regions>R)'. For example, a L-Complex (2R) configuration refers to two layers of the complex configuration connected through multiple randomly selected nodes. Each layer forms a region controlled by an agent. The network within

| Configuration | Algorithm | Time(s) | Total(Kb) | Max(Kb) | Optimality |
|---|---|---|---|---|---|
| L-Complex(1R) | DPOP | – | – | – | – |
| L-Complex(10R) | $D^2$ADP | 247.2 | 910.7 | 5.4 | 88.0% |
| L-Complex(15R) | $D^2$ADP | 389.4 | 1579.8 | 6.1 | 89.0% |
| L-Complex(20R) | $D^2$ADP | 516.6 | 2231.1 | 7.0 | 86.7% |
| L-Complex(25R) | $D^2$ADP | 540.0 | 2823.1 | 7.4 | 86.5% |
| L-Complex(30R) | $D^2$ADP | 786.0 | 16858.9 | 8.9 | 86.4% |

**Table 4: Solution quality, runtime and message size results for synthetic configurations**

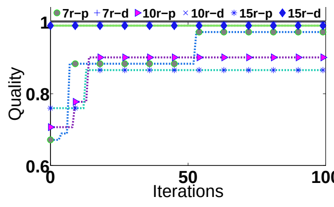
each layer is the same as 'Complex'. Layers are arranged in a 3D fashion with each layer connected via inter-layer edges to the layer above and below. Agents can only communicate via inter-layer edges. The number of inter-layer edges was about 10% of total edges (=14) in the 'Complex' network. Such a layered model of construction helps conserve the network structure of real power networks while increasing the scale with increasing number of layers. We consider instances with up to 30 layers. We consider each region to be one layer in the synthetic configurations. For instance, we use L-Complex(10R) to mean 10 layered 'Complex' configuration with 10 regions or 10 agents.

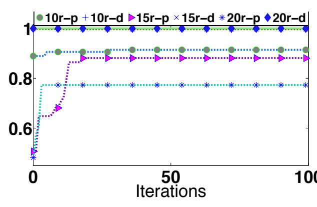
We have generated 10 random instances, where sink-weights are varied, for the synthetic configurations to ensure there is no specific dependence on sink weights. The capacity of power sources was set such that it was necessary for power to flow from one layer to another; one layer was severely deficient in total power, another had excess power. Table 4 shows the results on such synthetic instances. Even on such large and intricate instances, our decentralized approach is able to get good solution quality. Furthermore, the size of each individual message in our approach increases quite moderately w.r.t. increasing number of regions. This is because the messages only contain local information about the dual variables and the sub-gradient information for each cut edge. Thus, our approach scales well with the network size and is able to provide good quality solutions with limited message passing overhead.

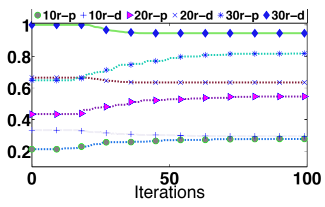
**Anytime Performance** We next show solution quality results obtained using $D^2$ADP on the 'Large' and 'Complex' configurations for different regions for each iteration. We show primal and dual values for each instance as the number of iterations is increased. Figure 3(a) provides the results on the 'Large' configuration. '7r-p' refers to the primal solution with 7 regions, '7r-d' refers to the dual solution with 7 regions, '15r-p' is primal for 15 regions and so on. We interpret the problem as a maximization problem with dual always providing an upper bound. We can clearly see from these results that our approach is able to provide good primal solutions even in early iterations. Figure 3(b) provides the results on the 'Complex' configuration with 10, 15 and 20 regions. Again, we get good solution quality for 10 and 15 regions. As the number of regions increased to 20, our approach had to relax and dualize many constraints. That resulted in performance hit. We do note that with 20 regions, each region has about 6 nodes. This represents a very harsh partitioning of the network. Still our approach was able to provide a decent solution quality.

The results for different L-Complex instances and configurations are presented in Figures 3(c) and 3(d). 10r-p in this graph refers to the primal quality for a 10 layered L-Complex configuration with 10 regions. We observe that as number of layers are increased, the duality gap increases. However, even on multiple instances of different L-Complex configurations, we obtain strong quality guarantees of around 85% or more of the optimal as shown in Figures 3(c) and 3(d).

**Comparison with centralized solver** As shown in table 4, existing decentralised algorithms are unable to scale to L-Complex


(a) 'Large'


(b) 'Complex'


(c) 'L-Complex' instance:1

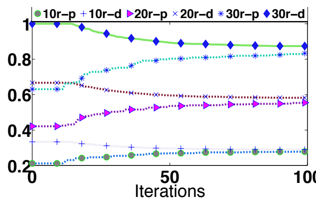
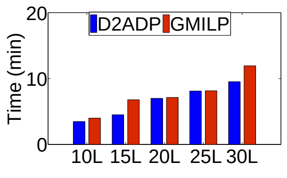

(d) 'L-Complex' instance:2

**Figure 3: Duality gap for real datasets (figures a & b) & synthetic datasets(figures c & d) with varying regions:#r-p denotes #regions-primal & #r-d denotes #regions-dual. All values are normalzied to 1**

instances. Hence, we benchmark the run-time performance of D$^2$ADP with the Global MILP (GMILP) of Table 2. This is not a fair comparison for our decentralized approach D$^2$ADP which solves the PSR problem using message passing, whereas the centralized solver has complete knowledge of the problem. Nonetheless, these results shed light on the effectiveness of our approach. Figure 4 provides the time taken by D$^2$ADP and GMILP for a given quality bound for different multilayered networks. We experiment with 10 random settings of sink-weights and compare the time taken for an average of result. Despite being a decentralized approach, D$^2$ADP runtime is highly competitive with respect to GMILP for a 85% quality bound. The best solution quality achieved (in percentage of optimal) for the respective L-Complex configuration by D$^2$ADP is mentioned on top of bars in figure 4(b). We show in figure 4(b) the time required by the GMILP solver (CPLEX) to achieve the same quality bound as provided by D$^2$ADP. This table further highlights that our approach is highly competitive to a strong centralized baseline. Thus, our key message from these results is that our approach can provide similar quality guarantees as a highly efficient centralized solver in a *decentralized* setting. In contrast, MILP solvers such as CPLEX are unable to work in a decentralized setting.

**Primal Extraction** Finally, we experiment with different values of constant $K$ to obtain the right setting for TBDP. A key practical issue with choosing $K$ according to the formula $0.5\beta \cdot (\min_j v_j)$ for a feeder tree is that such a value of $K$ that provides the theoretical guarantee for the FPTAS may be small, and unnecessarily lead to large running time for primal extraction. As also highlighted in [19], larger values of $K$ can provide similar empirical performance. Notice that in our case, the dual solution provides the upper bound, which is not affected by the primal extraction part. The theoretical guarantee for primal extraction does not affect the overall optimality gap for our approach.

We experiment on $K$ values as small as .01 and as large as 1. Figure 5(a) provides the time taken and Figure 5(b) provides the solution quality obtained for different values of $K$ on 3 different problem instances. We can see that for smaller value of $K$, runtime is significantly higher than for the larger $K$ values. However,


(a) 85% Quality

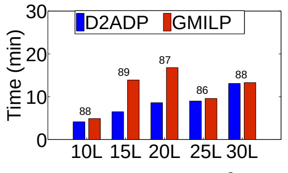
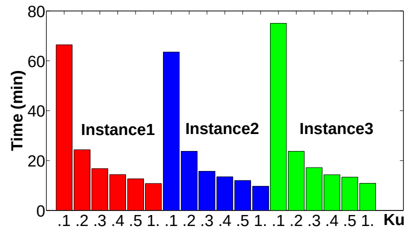

(b) Best Quality of D$^2$ADP

**Figure 4: Time comparision w.r.t. Quality with CPLEX**


(a) Time change w.r.t. $K$

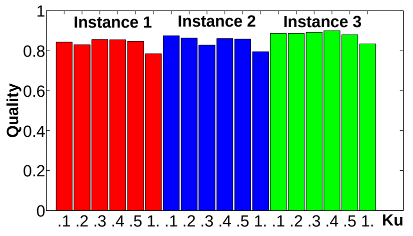

(b) Quality change w.r.t. $K$

**Figure 5: Time and Quality Variation w.r.t.$K$**

figure 5(b) shows that the solution quality remains almost the same for different $K$ values. Therefore, we chose $K = 0.4$ that provided the right tradeoff between time taken and optimality.

## 5. CONCLUSION

In our work, we addressed the problem of power grid reconfiguration after multiple line failures. We presented a number of advances for the multiagent version of the PSR problem. We developed a novel and iterative dual decomposition based approach that effectively utilized the underlying multi-region structure of the power grid. We also addressed the challenging problem of extracting a feasible solution providing anytime nature to our approach by developing a provable approximation technique. Our approach only requires local message-passing among different grid regions, resulting in a distributed approach. Using the quality bounds provided by our approach, we showed that it can achieve near-optimal solutions on a number of large real-world and synthetic benchmarks. Our approach is faster and significantly more scalable than the previous best multiagent approach for the PSR problem. We also showed empirically that our approach was highly competitive both in runtime and solution quality against a strong centralized baseline CPLEX, while retaining all the benefits of a decentralized approach.

## REFERENCES

[1] P. Bertoli, A. Cimatti, J. Slanley, and S. Thiebaux. Solving power supply restoration problems with planning via symbolic model checking. In *European Conference on Artificial Intelligence*, pages 576–580, Lyon, France, 2002.

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[3] C. Coffrin, P. Van Hentenryck, and R. Bent. Approximating line losses and apparent power in ac power flow linearizations. In *IEEE Power and Energy Society General Meeting*, pages 1–8, 2011.

[4] J. Griffin and S. Puller. *Electricity deregulation: choices and challenges*. University of Chicago Press, Chicago, 2005.

[5] T. Hadžic, A. Wasowski, and H. R. Andersen. Techniques for efficient interactive configuration of distribution networks. In *International joint conference on Artificial Intelligence*, pages 100–105, 2007.

[6] B. H. Kim and R. Baldick. Coarse-grained distributed optimal power flow. *IEEE Transactions on Power Systems*, 12(2):932–939, 1997.

[7] B. H. Kim and R. Baldick. A comparison of distributed optimal power flow algorithms. *Power Systems, IEEE Transactions on*, 15(2):599–604, 2000.

[8] J. Kok, M. Scheepers, and I. Kamphuis. Intelligence in electricity networks for embedding renewables and distributed generation. *Intelligent Infrastructures*, 42:179–209, 2010.

[9] A. Kumar, B. Faltings, and A. Petcu. Distributed constraint optimization with structured resource constraints. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 923–930, 2009.

[10] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 438–445, 2004.

[11] R. Mailler and V. R. Lesser. Asynchronous partial overlay: A new algorithm for solving distributed constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 25:529–576, 2006.

[12] T. Matsui and H. Matsuo. Considering equality on distributed constraint optimization problem for resource supply network. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 02*, pages 25–32, 2012.

[13] S. Miller, S. D. Ramchurn, and A. Rogers. Optimal decentralised dispatch of embedded generation in the smart grid. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, pages 281–288, 2012.

[14] T. Nagata and H. Sasaki. A multi-agent approach to power system restoration. *Power Systems, IEEE Transactions on*, 17(2):457–462, 2002.

[15] F. J. Nogales, F. J. Prieto, and A. J. Conejo. A decomposition methodology applied to the multi-area optimal power flow problem. *Annals of operations research*, 120(1-4):99–116, 2003.

[16] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. Putting the 'smarts' into the smart grid: A grand challenge for artificial intelligence. *Commun. ACM*, 55(4):86–97, Apr. 2012.

[17] S. Thiébaux, C. Coffrin, H. Hijazi, and J. K. Slaney. Planning with MIP for supply restoration in power distribution systems. In *International Joint Conference on Artificial Intelligence*, 2013.

[18] S. Thiebaux and M. Cordier. Supply restoration in power distribution system - a benchmark for planning under uncertainty. In *ECP*, pages 85–96, 2001.

[19] X. Wu, D. Sheldon, and S. Zilberstein. Rounded dynamic programming for tree-structured stochastic network design. In *AAAI Conference on Artificial Intelligence*, pages 479–485, 2014.