Optimal Decentralised Dispatch of Embedded Generation in the Smart Grid

Sam Miller, Sarvapali D. Ramchurn, Alex Rogers
Agents, Interaction and Complexity Group
School of Electronics and Computer Science
University of Southampton, UK
{sjom106,sdr,acr}@ecs.soton.ac.uk

ABSTRACT

Distribution network operators face a number of challenges; capacity constrained networks, and balancing electricity demand with generation from intermittent renewable resources. Thus, there is an increasing need for scalable approaches to facilitate optimal dispatch in the distribution network. To this end, we cast the optimal dispatch problem as a decentralised agent-based coordination problem and formalise it as a DCOP. We show how this can be decomposed as a factor graph and solved in a decentralised manner using algorithms based on the generalised distributive law; in particular, the max-sum algorithm. We go on to show that max-sum applied naïvely in this setting performs a large number of redundant computations. To address this issue, we present a novel decentralised message passing algorithm using dynamic programming that outperforms max-sum by pruning the search space. We empirically evaluate our algorithm using real distribution network data, showing that it outperforms (in terms of computational time and total size of messages sent) both a centralised approach, which uses IBM's ILOG CPLEX 12.2, and max-sum, for large networks.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*intelligent agents, multiagent systems*

General Terms

Algorithms, Theory, Performance

Keywords

DCOP, electricity, max-sum, coordination

1. INTRODUCTION

Due to recent incentives for cleaner electricity generation [13], there has been an increasing amount of renewable generators embedded in distribution networks [7, 9]. This poses a number of challenges for distribution network operators (DNOs). Firstly, electricity networks are already highly capacity constrained; adding additional generation that is not managed effectively may overload the networks [14]. Secondly, it is much harder to balance electricity demand with generation from intermittent renewable resources. If the DNO fails to balance the supply and demand, the network can potentially become unstable which may result in brownouts, and in the worst case, cascading blackouts.

Thus, there is a clear incentive for DNOs to implement optimal dispatch[1] methods that are able to address these issues. That is, how should the generators be coordinated, such that the cost of the network is minimised (i.e., in terms of carbon dioxide (CO_2) emissions or generator running expenditure), whilst satisfying the loads and network constraints. Coordinating generators with respect to network cost, is known as active network management (ANM), and has recently been addressed in the power systems community [3, 14].

Within the ANM domain, a number of authors address the issues of coordinating generation from intermittent resources in the transmission network (where lines are less constrained than in the distribution network) [2, 8]. For example, Davidson et al. present an algorithm for changing the power outputs of the generators in the transmission network such that the cost of the network is minimised [2]. However, their technique involves a central authority calculating each generator's power output; who must have all the information about the entire network in order to calculate an optimal solution. As the size of the network grows, solving an optimisation problem in a centralised manner eventually becomes infeasible due to the exponential nature of the constraints [6].

In contrast, others have attempted to improve upon centralised approaches by decomposing the optimal dispatch problem and distributing the computation of its solution in order to improve its scalability [8, 10, 16]. For example, Kim and Baldick introduce a decentralised algorithm which uses Lagrangian techniques [8]. However, their algorithm has only been tested on problems containing up to two regions. Thus, it is unclear whether this approach could be applied to a large network. In the multi-agent systems literature, Kumar et al. introduce a message passing technique which extends distributed pseudotree optimisation procedure (DPOP) to solve the related area of research for reconfiguring feeder trees within a distribution network [10]. While this approach is decentralised, and was shown to work

[1]Optimal dispatch involves coordinating generators such that the loads and constraints of the network are satisfied.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems – Innovative Applications Track (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

on realistic sized networks, it does not address the problems highlighted above of incorporating an increasing amount of distributed generators (DGs) in the distribution network, and the need to coordinate their output.

Vytelingum et al. tackle the optimal dispatch problem by managing the trading of electricity between nodes on a network [16]. However, their technique has only been tested on problems containing up to 16 nodes. Thus, again it is unclear whether this approach could be applied to a larger network. Moreover, their technique is partially centralised; since each agent needs to know the entire network topology. In a large network, maintaining this system-wide knowledge is problematic, especially when faced with renewable generators whose output is continuously changing.

Against this background, in this paper we address the challenge of coordinating large numbers of renewable generators, embedded in the distribution network, by decomposing the optimal dispatch problem into a decentralised agent-based coordination problem; represented as a distributed constraint optimisation problem (DCOP). In more detail, each node in the network is represented by an agent which undertakes some of the computation required to solve the optimal dispatch problem; such that demands within the network are satisfied and CO_2 emissions of the entire network are minimised. We further decompose the DCOP as a factor graph and solve in a decentralised manner using algorithms based on the generalised distributive law (GDL) [1]; in particular, the max-sum algorithm [4]. We go on to show that max-sum applied naïvely in this setting performs a large number of redundant computations.

To address this issue, we present a novel message passing algorithm, called DYDOP (DYnamic programming Decentralised OPtimal dispatch), to calculate an optimal solution in a decentralised fashion. In particular, we solve the optimal dispatch problem on the most common distribution network types, namely radial networks, which tend to incorporate a high number of branches and sources [17]; for which centralised solutions scale poorly. Other common types of distribution networks include interconnected networks,[2] typically found in urban settings [5]. However, relays throughout the network ensure that all but one path is active at any one time; Multiple paths are present for security of supply. Therefore, our assumption of acyclic distribution networks throughout this paper is fully justified. Crucially, our algorithm handles the complexities of balancing flows within the network, *without needing central verification* of a particular solution. Thus, this paper makes the following contributions to the state of the art:

1. We provide a new formalism of the optimal dispatch problem as a DCOP. We show how this can be decomposed as a factor graph and solved using algorithms based on the GDL family, such as max-sum.

2. We present DYDOP, a novel decentralised message passing algorithm, that outperforms max-sum by only exploring the search space of feasible generator and distribution cable states.

3. We provide proof of the optimality of our algorithm and empirically evaluate it, on a large distribution network in India, showing that it outperforms (in terms of computational time and total size of messages sent) both a highly optimised centralised approach, which uses IBM's ILOG CPLEX 12.2, and max-sum.

[2]In an interconnected network there are multiple paths from a substation to a load.

When taken all together, our results set the benchmarks for the deployment of agent-based coordination algorithms to solve the optimal dispatch problem in the smart grid.

The remainder of this paper is organised as follows: Section 2 presents the formal model of the electricity network used for optimal dispatch. Section 3 details a new formalism of the optimal dispatch problem as a DCOP, and Section 4 shows how it can be solved by using max-sum. Section 5 presents our novel decentralised message passing algorithm DYDOP, presenting proof of optimality. Section 6 gives an empirical evaluation of DYDOP and Section 7 provides a discussion for future work. Finally, Section 8 concludes.

2. ELECTRICITY NETWORK MODEL

We now formally describe the model of an electricity network for which we need to solve the optimal dispatch problem. Hence, we consider the electric distribution network to be a network of generators, loads and distribution cables.

In more detail, we consider a set of n generators $\mathbf{G} = \{g_1, ..., g_n\}$. Each generator g_i has a certain discrete power output variable $\alpha_i \in S_i$ kW, where $S_i = \{s^i_1, ..., s^i_{q_i}\}$, $s^i_{q_i} \in \mathbb{R}^+$, $q_i \in \mathbb{Z}^+$ is the number of power output values for generator g_i, and $\mathbf{S}$ is an n-ary Cartesian product of S_i such that $\mathbf{S} = \{S_i \times ... \times S_n\}$. Let $\boldsymbol{\alpha} \in \mathbf{S}$ denote the set of power output variables for the generators in $\mathbf{G}$. Let $e_i = CI_i\alpha_i$ denote the CO_2 emissions that are produced when g_i, with carbon intensity $CI_i \in \mathbb{R}^+\text{kgCO}_2/\text{kWh}$, outputs α_i.

We consider a set of m loads $\mathbf{L} = \{l_1, ..., l_m\}$. Each load l_i has a certain power consumption $\beta_i \in \mathbb{R}^-$ kW, where $\boldsymbol{\beta} = \{\beta_1, ..., \beta_m\}$ is the set of power consumption variables for the loads in $\mathbf{L}$.

We denote $\mathbf{V} = \{v_1, ..., v_k\}$ as the set of k nodes within the network. A node relays power to other nodes but can also contain a combination of generators and loads. Let $adj(v_i)$ denote all nodes that are connected to v_i via a distribution cable, let $\mathbf{L}(v_i)$ be the set of loads that are at v_i and $\mathbf{G}(v_i)$ be the set of generators that are at v_i.

$\mathbf{T}$ is the set of s distribution cables within the network, where $t_{ij} \in \mathbf{T}$ denotes the distribution cable between v_i and v_j. Each distribution cable has an associated thermal capacity $t^c_{ij} \in \mathbb{R}^+$ kW, which is the maximum power the cable can safely carry. It should be noted that we assume that all the distribution cables have the same reactance.

Finally, $\mathbf{W}(\mathbf{V}, \mathbf{T})$ is a finite undirected graph describing a network of nodes and distribution cables. $\mathbf{F}$ is the set of all power flows $f_{ij} \in \mathbb{R}$ kW, along the distribution cables in the network. Given the above definitions, the optimal dispatch problem, of finding an allocation of power outputs $\boldsymbol{\alpha}$, is defined as the problem of minimising:

$$\arg\min_{\boldsymbol{\alpha}} \sum_{i=0}^{n} CI_i\alpha_i \tag{1}$$

subject to the following constraints:

Constraint 1 The flow along a distribution cable cannot exceed its capacity:

$$|f_{ij}| \leq t^c_{ij} \tag{2}$$

Constraint 2 The net flow from v_i to v_j must be the opposite of the net flow from v_j to v_i:

$$f_{ij} = -f_{ji} \tag{3}$$

Constraint 3 The sum of the generators at v_i, the sum of the loads at v_i and the net flow from all nodes w connected to v_i is zero:

$$\sum_{w \in adj(v_i)} f_{wi} + \sum_{l \in \mathbf{L}(v_i)} \beta_l + \sum_{g \in \mathbf{G}(v_i)} \alpha_g = 0 \tag{4}$$

Having presented a model of the electricity network, the following section decomposes the problem into a DCOP.

3. DCOP REPRESENTATION

Formally, a DCOP is a tuple $\langle \mathbf{X}, \mathbf{D}, \mathbf{U} \rangle$ consisting of a set of h variables $\mathbf{X} = \{x_1, ..., x_h\}$ which can be assigned discrete values in the set of finite domains $\mathbf{D} = \{\mathbf{d}_1, ..., \mathbf{d}_h\}$ respectively. In our representation, $\mathbf{X} = \{\boldsymbol{\alpha}, \mathbf{F}\}$, with the domain:

$$\mathbf{d}_i = \begin{cases} S_i & \text{when } x_i \text{ is a generator} \\ & \text{when } x_i \text{ corresponds to the} \\ \{-t^c_{ab}, ..., t^c_{ab}\} & \text{distribution cable } t_{ab} \\ & \text{between } v_a \text{ and } v_b \end{cases} \tag{5}$$

We note the set of relations as $\mathbf{U} = \{U_1, ..., U_k\}$ (also called utilities) where $U_i : \mathbf{d}_{i1} \times ... \times \mathbf{d}_{ij} \rightarrow \mathbb{R}^+$ denotes the cost of each possible combination of the involved variable values. We denote $\mathbf{A}$ to be the set of k agents. Each variable is assigned to an agent. Only the agent who is assigned the variable has knowledge of its domain and control over its value. Moreover, the utility U_i corresponds to the utility of agent i. In the context of the electricity network, U_i maps to the CO_2 emissions of v_i with respect to the constraints of the network (i.e., a lower cost means lower CO_2 emissions):

$$U_i = \begin{cases} \sum_{g \in \mathbf{G}(v_i)} CI_g \alpha_g & \text{if Equation (4) holds for } v_i \\ \infty & \text{otherwise} \end{cases} \tag{6}$$

where ∞ is used to penalise states that lead to inconsistent flows within the network (i.e., states that do not satisfy Equation 4).

With this in mind, it can be seen that the optimisation function for the electricity network, Equation (1), can be factorised in terms of the agent utility functions using Equation (6). The goal of the agents is to find an assignment $\mathbf{X}^*$ for the variables in $\mathbf{X}$ that minimises the sum of the costs:

$$\arg\min_{\mathbf{X}^*} \sum_{i=0}^{k} U_i \tag{7}$$

Typically, a DCOP can be represented by a factor graph, whose vertices correspond to variables and the edges denote the dependencies between the variables (i.e., the utility functions). Crucially, we provide a mapping of the DCOP to a factor graph that preserves the acyclic topology of the electricity network. Moreover, this mapping balances all of loads with generation, whilst satisfying the flow constraints of each distribution cable, and the constraints of the generators, in a fully decentralised way without needing centralised verification. Figure 1(a) shows an electricity distribution network consisting of distribution cables, generators and nodes. Example values for generator's maximum output, distribution cable's thermal capacity and power consumption at the loads are given. Node v_0 is connected to the rest of the electricity grid. Figure 1(b) shows the corresponding factor graph. By decomposing into a factor graph, the optimal dispatch problem can be solved using an algorithm from the GDL family, such as max-sum.

We choose max-sum to solve the DCOP because it maps directly onto a factor graph, and directly works with n-ary constraints (i.e., functions connected to more than two variables, see U_5 on Figure 1(b) for an example) without any additional modifications. Other algorithms which transform the DCOP into a depth first search (DFS) tree, such as ADOPT [11] and DPOP [12], suffer from scaling issues with the height and the width of the DFS tree respectively. Thus, max-sum is a natural fit to the optimal dispatch problem in a distribution network because networks of this nature often contain a large number of nodes and branches.

In max-sum, functions and variables can be arbitrarily assigned to any agent. However, in our model, each agent is assigned to compute one function which is associated to a specific node within the network. Moreover, a natural assignment of variables to agents involves an agent controlling the generator variables at its designated node, and the distribution cable variables connected to its node. If two or more agent's functions share the same variable, the variable is arbitrarily assigned to one of them. In Figure 1(b), the dashed circles give an example of the agents.

More importantly, since max-sum has been proven to converge to an optimal solution on acyclic factor graphs, and given that we provide a mapping from an acyclic electricity network to an acyclic factor graph, max-sum will be able to calculate the optimum solution to the optimal dispatch problem. The following section introduces the max-sum algorithm and explains how it can be applied to an electricity network.

4. MAX-SUM OPTIMAL DISPATCH

The max-sum algorithm (or min-sum as is the case with minimising CO_2 emissions) uses message passing in order to propagate the utilities of the variables around the factor graph. Messages are sent from variable to function, and from function to variable:

From variable to function:

$$Q_{b \rightarrow a}(x_b) = \sum_{a' \in A(b) \backslash a} R_{a' \rightarrow b}(x_b) \tag{8}$$

From function to variable:

$$R_{a \rightarrow b}(x_b) = \min_{\mathbf{X}_a \backslash b} \left[U_a(\mathbf{X}_a) + \sum_{b' \in B(a) \backslash b} Q_{b' \rightarrow a}(x_{b'}) \right] \tag{9}$$

where $B(a)$ is the set of variables connected to the function a, $A(b)$ is the set of functions connected to the variable x_b, and finally $\mathbf{X}_a \backslash b \equiv \{x_{b'} : b' \in B(a) \backslash b\}$.

A max-sum message being sent from function to distribution cable variable is a function of the flow in the cable with its domain bounded by the thermal capacity of the distribution cable. Consider the following example, shown in Figure 1(a). Let the distribution cable t_{59} between v_5 and v_9 have a thermal capacity t^c_{59} of 40kW, the load l_9 at v_3

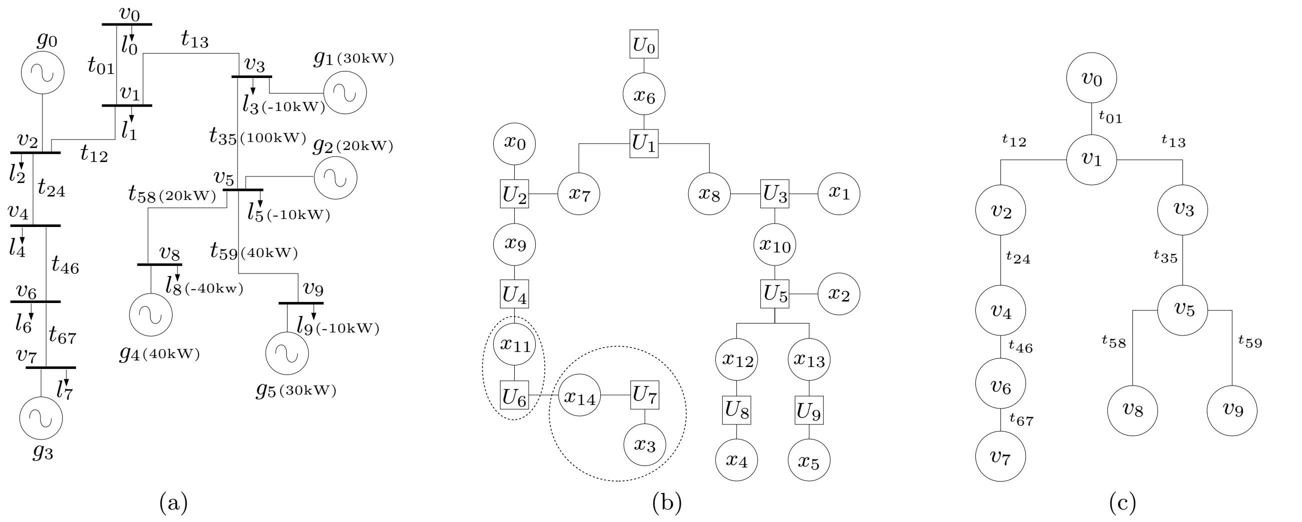

Figure 1: (a) An electricity distribution network. Showing example values for generator's maximum output, distribution cable's thermal capacity and power consumption at the loads. Node v_0 is connected to the rest of the electricity grid. (b) A factor graph representation of the same network. (c) The tree representation used by DYDOP.

be -10kW, and the generator g_5 at v_9 have a maximum output of 30kW. The message $R_{5\to13}(x_{13})$, sent from U_5 to x_{13} on the corresponding factor graph, Figure 1(b), will have domain $x_{13} \in \{-40, ..., 0, ..., 40\}$, having 81 utility values corresponding to the 81 variable states, discretised by 1kW steps. A +ve variable state indicates that the flow f_{59} is traveling from v_5 to v_9, and a -ve variable state indicates that f_{59} is traveling from v_9 to v_5.

A max-sum message being sent from function to generator variable is bounded by the maximum output of the generator. Consider the following example. Let the generator g_1 at v_3 have a maximum output of 30kW. The message $R_{3\to1}(x_1)$ will have domain $x_1 \in \{0, ..., 30\}$, having 31 utility values corresponding to the 31 variable states. Each state indicates the amount of power g_1 is producing α_1.

Messages are propagated around the factor graph until the values of the messages converge. Messages are guaranteed to converge to the optimal solution on acyclic graphs. At which point each variable chooses its optimal state based on the sum of the messages it has received:

$$Z_b(x_b) = \sum_{a \in A(b)} R_{a\to b}(x_b) \tag{10}$$

However, simply applying the max-sum algorithm naïvely in this manner produces poor performance. This is because much of the search space is infeasible and does not need to be searched. For instance, consider the previous example for the message $R_{5\to13}(x_{13})$. The message has a total of 81 variable states. However, the maximum amount of power that could travel along t_{59} from v_5 to v_9, in order to satisfy l_9, is only 10kW. Moreover, the maximum output of g_5 means that the maximum amount of power that could travel along t_{59} from v_9 to v_5, after l_9 is satisfied, is 20kW. Therefore, the utilities calculated for variable states $\{-40, ..., -21\}$ and $\{11, ..., 40\}$ are all infeasible. This highlights the wasted computation that a naïve implementation of max-sum performs. The domain of the message is bounded by t^c_{59}. However, the actual feasible states are dependant on the load and the available generation at v_9, which is considerably less. As the network size grows, this wasted computation becomes a major overhead (as we show in Section 5.1.2).

Thus, to address this issue, we present a novel decentralised message passing algorithm, DYDOP, which propagates messages from leaf nodes to the root of the tree network, such that only the utility of feasible states are calculated. As we show later, doing so greatly reduces the computation time as it allows us to prune much of the search space.

5. DYDOP OPTIMAL DISPATCH

We represent an acyclic electricity network as an acyclic network of nodes connected by distribution cables; Figure 1(c) shows the electricity network in Figure 1(a) transformed into this representation. DYDOP is applied to the acyclic network and uses a dynamic programming approach. Each node, which is controlled by an agent, has exactly one parent node and zero or more child nodes, apart from one node v_0 which is the root node and has no parent. Leaf nodes have no children, v_7, v_8 and v_9. Each node is assumed to have one or more generators, each with an associated carbon intensity, and one or more loads. DYDOP proceeds in two phases (which we describe in more detail in the following section):

Phase 1 – Value Calculation *PowerCost* messages are sent from the leaf nodes to the root node. A node waits until it has received *PowerCost* messages from all of its children before computing its own *PowerCost* message which it sends to its parent. Each *PowerCost* message describes the CO_2 emissions of its own generation and the generation of its children.

Phase 2 – Value Propagation When the root node receives *PowerCost* messages from all of its children, it calculates its optimum power output such that all the demands of the network are satisfied and the CO_2 emissions are minimised. It then propagates power flow values to all its children which in turn propagate power flow values to their children.

The algorithm terminates when all leaf nodes receive a power flow value, at which point each node knows exactly what power it needs to output. We elaborate on the two phases below.

5.1 Phase 1: Value Calculation

In what follows we give a detailed overview of the DYDOP's value calculation phase. Section 5.1.1 introduces the structure of a *PowerCost* message, Section 5.1.2 describes how a leaf node constructs its *PowerCost* messages, and finally Section 5.1.3 details how a node merges its children's *PowerCost* messages.

5.1.1 *PowerCost* Messages

A *PowerCost* message sent from v_i to its parent $\hat{v_i}$, is an array of y *flowCO* elements:

$$PowerCost_{i \to \hat{i}} = [flowCO_1, ..., flowCO_y] \tag{11}$$

A *flowCO* element describes the CO_2 emissions that occur, when v_i and all of its children $chi(v_i)$ output certain amounts of power, such that there is a specified flow of power between v_i and its parent $\hat{v_i}$ along the distribution cable $t_{i\hat{i}}$:

$$flowCo_j = < f_{i\hat{i}}, \boldsymbol{\gamma}(f_{i\hat{i}}) > \tag{12}$$

where $f_{i\hat{i}} \in \mathbb{Z}$ kW is the resultant power flow travelling along $t_{i\hat{i}}$, and $|f_{i\hat{i}}| \leq t^c_{i\hat{i}}$ where $t^c_{i\hat{i}}$ is the thermal capacity of $t_{i\hat{i}}$. Note that $f_{i\hat{i}} > 0$ denotes the resulting power is flowing out of v_i to $\hat{v_i}$, $f_{i\hat{i}} < 0$ denotes the resulting power is flowing into v_i from $\hat{v_i}$, $f_{i\hat{i}} = 0$ denotes no power is flowing between v_i and $\hat{v_i}$. The function $\boldsymbol{\gamma} : \mathbb{R} \to \mathbb{R}^+$ $kgCO_2$/h denotes the CO_2 emissions that result from v_i and all of its children generating certain amounts of power.[3] Each *flowCO* element that v_i calculates maps to an *OPCState* which describes v_i's power output along with the *flowCO* elements of each of its children that results in the CO_2 emission described by the function $\boldsymbol{\gamma}$. This mapping represents the dynamic programming aspect of DYDOP since as power flow values are propagated down the tree, during the value propagation phase, the associated *OPCState* is used to find node v_i's power output given a particular power flow $f_{i\hat{i}}$.

5.1.2 Constructing a *PowerCost* Message at a leaf

Only the leaf node's power output needs to be taken into consideration when a leaf *PowerCost* message is constructed. For each power output v_i can produce, it constructs a corresponding *flowCO* element with flow $f_{i\hat{i}}$ calculated as:

$$f_{i\hat{i}} = \sum_{l \in \mathbf{L}(v_i)} \beta_l + \sum_{g \in \mathbf{G}(v_i)} \alpha_g \tag{13}$$

giving the resultant power flowing between v_i and $\hat{v_i}$. The CO_2 emissions $\boldsymbol{\gamma}$ of the *flowCO* element, is calculated as:

$$\boldsymbol{\gamma}(f_{i\hat{i}}) = \sum_{g \in \mathbf{G}(v_i)} \alpha_g CI_g \tag{14}$$

[3]Node v_9, Figure 1(c), with a carbon intensity of 0.3$kgCO_2$/kWh and a power output of 20kW, will have a resulting CO_2 emissions of 6$kgCO_2$/h and 10kW of resulting power travelling to v_5.

Algorithm 1 Constructing a leaf node *PowerCost* message

```
1. for αi ← 0 to genMax {
2.    rFlow ← αi + βi;
3.    rCO ← αi * CIi;
4.    flowCO(rFlow, rCO);
5.    linkToOPCState(flowCO, αi);
6. }
7. sendPowerCostMessageToParent();
```

where CI_g is the carbon intensity of generator g situated at v_i. See Algorithm 1 for a pseudocode representation of constructing a leaf node *PowerCost* message. We iterate through the generators different outputs, up to its maximum (line 1). For each output the resultant flow is calculated, (line 2) and the corresponding CO_2 emissions, (line 3). A *flowCO* element is created, (line 4), and then linked to the generators output which resulted in the resultant CO_2 emissions, (line 5). All the *flowCO* elements created are added to a *PowerCost* message and then sent to the nodes parent, (line 7). Note that the *OPCState*'s that are linked to by each *flowCO* element are never sent on to the parent node and are instead kept for use during phase 2 of the algorithm.

Consider the following $PowerCost_{9 \to 5}$ message, which v_9 sends to v_5, see Figure 1(a). Let the distribution cable t_{59} have a thermal capacity t^c_{59} of 40kW, the load l_9 be -10kW, the generator g_5 have a maximum output of 30kW and g_5 have a carbon intensity CI_5 of 0.1$kgCO_2$/kWh. The following is part of the $PowerCost_{9 \to 5}$ message:

$$\begin{array}{lcccc} flowCo_j & = & < 0, 1.0 > & \to & [+10kW] \\ flowCo_{j+1} & = & < 1, 1.1 > & \to & [+11kW] \\ flowCo_{j+2} & = & < 2, 1.2 > & \to & [+12kW] \end{array}$$

Now, $flowCo_{j+2}$ indicates that a flow 2kW, from v_9 to v_5, will result in 1.2$kgCO_2$ emission with g_5 outputting 12kW. The total number of *flowCO* elements in $PowerCost_{9 \to 5}$ is 31. By contrast, compare with the example $R_{5 \to 13}(x_{13})$ message in Section 4, which has 81 variable states instead. This further highlights the wasted computation that the naïve implementation of max-sum performs.

5.1.3 Merging *PowerCost* messages

For each v_i that has at least one child, the *PowerCost* messages that it receives must be processed in order to produce its own *PowerCost* message that it sends to $\hat{v_i}$. The amount of power that can flow from v_i to $\hat{v_i}$, or from $\hat{v_i}$ to v_i, is bounded by $t^c_{i\hat{i}}$. With these bounds, v_i is able to calculate each valid flow that can travel into or out of it. For each valid flow, v_i calculates the minimum CO_2 emissions that result from v_i's output, and all of its children's outputs. To calculate the *flowCO* element for each resultant flow with the lowest CO_2 emissions value, v_i iterates through every possible power output that it can produce and every *flowCO* element from each of its children's *PowerCost* messages. A state represents the combination of one *flowCO* element from each of its children and v_i's power output. The flow $f_{i\hat{i}}$ of this state is calculated as:

$$f_{i\hat{i}} = \sum_{l \in \mathbf{L}(v_i)} \beta_l + \sum_{g \in \mathbf{G}(v_i)} \alpha_g + \sum_{c \in chi(v_i)} f_{ci} \tag{15}$$

Algorithm 2 Merging *PowerCost* messages

```
1. for αi ← 0 to genMax {
2.   foreach childPowerCost {
3.     rFlow ← αi + load + sum(OPCState);
4.     rCO ← (αi * CIi) + sum(OPCState);
5.     if(min(rFlow, rCO)) {
6.       PowerCost(rFlow, rCO);
7.       setNewMinimum(PowerCost);
8.       linkToOPCState(PowerCost, αi);
9.     }
10.  }
11.}
12.sendPowerCostMessageToParent();
```

where $\sum_{c \in chi(v_i)} f_{ci}$ is the sum of the chosen *flowCO* elements' flows from each of v_i's children. In order to choose the minimum state for each resultant flow, the CO_2 emissions of the state must be calculated as follows:

$$\gamma(f_{i\hat{i}}) = \sum_{g \in \mathbf{G}(v_i)} \alpha_g CI_g + \sum_{c \in chi(v_i)} \gamma(f_{ci}) \tag{16}$$

where $\sum_{c \in chi(v_i)} \gamma(f_{ci})$ is the sum of the chosen *flowCO* elements' CO_2 emissions from each of v_i's children. See Algorithm 2 for a pseudocode representation of merging *PowerCost* messages. We iterate through the generators different outputs, up to its maximum (line 1). For each output, we iterate through every possible combination of the *flowCO* elements from each of the its children's *PowerCost* messages (line 2). For a particular *OPCState* (i.e., a combination of *flowCO* elements, one from each child, and the generators output) the resultant flow is calculated by summing each flow of the *flowCO* elements, in the *OPCState*, with the generator output and the load, (line 3). Similarly, the resultant CO_2 emission is calculated by summing each CO_2 emission of the *flowCO* elements, in the *OPCState*, together with the product of the generators output and its carbon intensity, (line 4). If the resultant CO_2 emissions is the minimum recorded for the particular resultant flow, (line 5), then the *flowCO* element is created, (line 6), and set as the new minimum for that particular resultant flow, (line 7). The *flowCO* element is linked to the *OPCState*, (line 8). All the *flowCO* elements created are added to a *PowerCost* message and then sent to the nodes parent, (line 12).

As an example of merging *PowerCost* messages, consider the following $PowerCost_{5\rightarrow3}$ message, v_5 sends to v_3, see Figure 1(a). Let t^c_{35} be 100kW, t^c_{58} be 20kW, t^c_{59} be 40kW, l_5 be -10kW, l_8 be -40kW, l_9 be -10kW, g_2 have maximum output 20kW, CI_2 be 0.7kgCO_2/kWh, g_4 have maximum output 40kW, CI_4 be 0.25kgCO_2/kWh, g_5 have maximum output 30kW and CI_5 be 0.1kgCO_2/kWh. The following is part of the $PowerCost_{5\rightarrow3}$ message (after receiving messages from v_8 and v_9):

$$\begin{array}{lcccc} flowCO_j & = & < -10, 8 > & \rightarrow & [+0kW]8(-20)9(20) \\ flowCo_{j+1} & = & < -9, 8.25 > & \rightarrow & [+0kW]8(-19)9(20) \\ flowCo_{j+2} & = & < -8, 8.5 > & \rightarrow & [+0kW]8(-18)9(20) \end{array}$$

Now, $flowCo_{j+1}$ indicates that a flow of 9kW, from v_3 to v_5, will result in 8.25kgCO_2 emission with g_2 outputting 0kW, a flow 19kW from v_5 to v_8, and a flow 20kW from v_9 and v_5. The following section describes the second phase of DYDOP whereby power output values are propagated from the root node to the leaf nodes.

5.2 Phase 2: Value Propagation

Once the root node has received *PowerCost* messages from all of its children, it calculates how much power to output in order to satisfy all the loads within the network and minimise CO_2 emissions. It does this by iterating through every possible power output that it can produce and every *flowCO* element from each of its children's *PowerCost* messages. Equation (15) is used to calculate the resultant flow of a state. If the flow is not equal to zero, then this particular state for the network is infeasible, since excess flow means that supply and demand is imbalanced. For every state that has a flow equal to zero, the CO_2 emissions of the network are calculated by using equation (16).

The state with the minimum CO_2 emissions is selected as the optimum state of the network. Power flow values are then sent to each of the root node's children telling them which of their *flowCO* elements resulted in the minimum CO_2 emission. The child retrieves the correct *flowCO* element by matching the power flow value sent to them with the flow from the *flowCO* message. The *OPCState* which is referenced by each child recipient's corresponding *flowCO* element tells the child exactly how much power to output. The child recipient can then send the power flow of each *flowCO* element specified in the *OPCState* to each of its corresponding children. Power flow values are propagated in this manner to the leaf nodes, at which point each node in the network knows their optimum power output that results in the minimum CO_2 emissions for the entire network.

5.3 Completeness and Correctness

In what follows, we prove that DYDOP applied to trees is complete and correct:

PROPOSITION 1. *DYDOP is complete*

PROOF. *To construct PowerCost messages, v_i must iterate through all of its own possible generator outputs, S_i, and every flowCO element from each of its children's PowerCost messages. Each flowCO element contains the minimum CO_2 emissions that results from each $l \in \boldsymbol{L}(v_i)$, and all of its children's loads, being satisfied. The root node chooses a feasible state that results in the minimum CO_2 emissions. Therefore, at each node, all feasible states are evaluated and the root node chooses the optimal state which minimises CO_2. Hence, the algorithm is complete.* □

PROPOSITION 2. *DYDOP is correct*

PROOF. *This proof follows on from proposition 1. When constructing messages, v_i only evaluates feasible states; the states that conform to equations (2) – (4) and each $g \in \boldsymbol{G}(v_i)$'s maximum power output. Each message will contain the minimum CO_2 emissions that result from a feasible set of states. Therefore, any solution calculated by the algorithm will be valid as it has explicitly conformed to the constraints of the entire network. Hence, the algorithm is correct.* □

5.4 Computational Complexity

Here, the worst-case complexity of DYDOP is calculated, with regards to the size of the network and the number of children a node has, in order to show its suitability for large optimal dispatch problems.

PROPOSITION 3. *The size of PowerCost messages that are sent by DYDOP grows linearly with the size of the network*

PROOF. *In the worst case, the maximum size of the message v_i has to create and send to $\hat{v}_i$ is Φ_i:*

$$\Phi_i = \frac{2t^c_{i\hat{i}}}{X_{\alpha_i}} \tag{17}$$

where $X_{\alpha_i} \in \mathbb{Z}^+$ is the discretisation of α_i and is currently 1; since each generator can produce power in 1 unit intervals. If generators are restricted to produce power in greater intervals, the size of the messages sent by each node can be reduced. In the worst case, the size of the messages DYDOP has to create and send in total is:

$$\sum_{v_i \in \boldsymbol{V} \setminus v_r} \Phi_i \tag{18}$$

where v_r is the root node. Therefore, the size of the messages DYDOP sends grows linearly in $O(|\boldsymbol{V}|)$. $\square$

PROPOSITION 4. *The number of states that v_i must iterate through is exponential with $|chi_i|$*

PROOF. *When merging PowerCost messages, v_i must iterate through all states in the Cartesian product of all of its children's states and its own power output values. Therefore, the number of states a node must iterate through in the worst case grows exponentially in $O(M^{cmax})$, where cmax is the number of children a node has and M is the number of states a child has with a discretisation of X_{α_i}.* $\square$

Even though the worst-case complexity of DYDOP is exponential in the number of children a node has, this is significantly less than the total number of nodes in the entire network. Thus, DYDOP may be able to exploit the structure of the network, unlike a centralised algorithm that does not explicitly take this structure into consideration, and compute an optimal solution faster. Therefore, the following section empirically evaluates DYDOP against a centralised approach and max-sum.

6. EMPIRICAL EVALUATION

In order to empirically evaluate DYDOP against max-sum and a centralised approach, we conducted an experiment on a real distribution network. The distribution network used is located in India and contains 76 substations;[4] the majority of the substations can further be connected to as many as 400 nodes. We only use one network because the topologies of distribution networks are largely similar. Our experiment was run in Java on a 2.67GHz Intel Xeon quadcore with 12GB of RAM, and was set up as follows. The number of additional nodes that could be connected to each substation was varied from 1 to 14, each with 50 iterations. During each iteration, nodes are assigned uniformly random loads, generators and carbon intensities. Each generator has 10 discrete power output levels and each distribution cable has its specified thermal capacity.

[4]A substation connects several distribution cables together and may contain generators, loads or transformers.

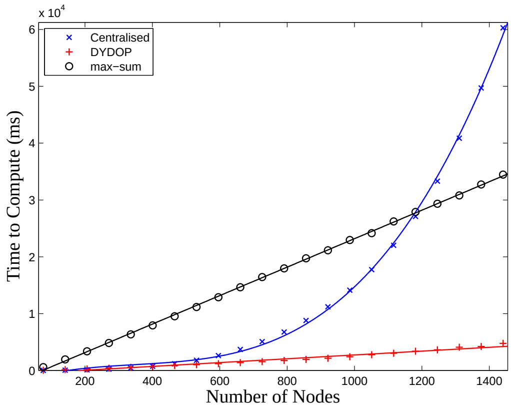

Figure 2: Time to compute a solution. India distribution network, 76 substations, varied number of nodes at each substation.

Figure 2 shows the computation time for the centralised approach, max-sum and DYDOP (error bars omitted due to being negligible). It can be seen that to start with, the centralised approach is as fast at computing a solution compared with DYDOP. However, after a network size of 460 nodes (which equates to only 6 additional nodes at each substation) DYDOP becomes significantly faster at computing a solution compared to the centralised approach. We used IBM's ILOG CPLEX 12.2 for the centralised approach which is highly optimised for solving optimisation problems.

Both DYDOP and max-sum's computation times increase linearly with the size of the network. This is because they both exploit the topology of the network. However, max-sum's computation time sharply increases compared with DYDOP. This highlights the unnecessary computation that max-sum is performing for infeasible variable states and shows the advantage of DYDOP. This is further highlighted in Figure 3 which shows that the total size of the messages sent using max-sum is much higher than DYDOP. Max-sum sends twice as many messages as DYDOP for the largest number of nodes we tested.

In contrast, the centralised computation time increases exponentially with the size of the network because it is unaware of the network structure, and seeks to solve the combinatorial optimisation by more standard approaches, such as the simplex method. Thus, as more DGs are added to distribution networks, it is clear that a centralised approach will quickly take an infeasible amount of time to compute a solution to the optimal dispatch problem.

In comparison, DYDOP is able to handle distribution networks with a large number of DGs and still calculate a solution in linear time. Therefore, our algorithm is a very good candidate for DNOs to use when solving future optimal dispatch problems in the ever growing distribution networks.

7. DISCUSSION

We believe DYDOP can be readily applied in many real-world electricity networks given the speed at which it resolves the generator outputs and the small amount of communication it requires. Particular applications include micro-grids with large numbers of small solar panels or micro-storage devices (on University campuses or military bases).

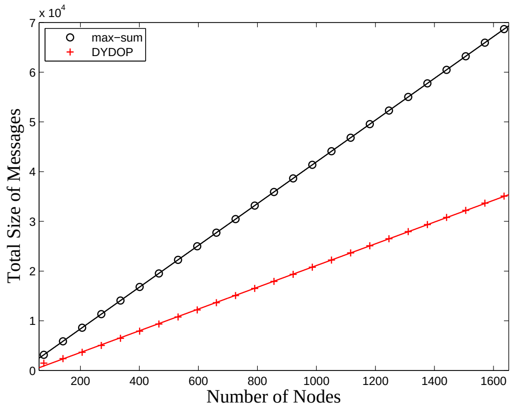

Figure 3: Sum total messages sent. India distribution network, 76 substations, varied number of nodes at each substation.

These applications typically involve network topologies that are either trees or radial and therefore match the type of network that DYDOP works on. Moreover, since the generators in these settings are typically low-power and discretise their power outputs (e.g. solar panels and batteries typically have set power outputs and can either be on or off), the assumptions we make about discretised generator outputs is perfectly valid in such settings.

Generalising our work to settings with non-discrete generator outputs will instead require handling continuous variables within DYDOP and it may be possible to extend some of the techniques introduced by [15] to do so. Moreover, to consider other distribution network topologies such as ring main,[5] we believe [18] can act as a starting point as they show that GDL algorithms can be made to converge on networks with a single loop.

8. CONCLUSIONS

In this paper we addressed the optimal dispatch challenges faced by DNOs. Namely how an increasing amount of cleaner DGs can be added to already highly constrained distribution networks, and coordinated in an efficient fashion using optimal dispatch. We provided a DCOP formulation of the optimal dispatch problem; we showed how this can be decomposed as a factor graph and solved in a decentralised manner using algorithms based on GDL; in particular, the max-sum algorithm. Furthermore, we showed that max-sum applied naïvely in this setting performs a large number of redundant computations.

To address this issue, we presented DYDOP, a novel decentralised message passing algorithm using dynamic programming, that outperforms max-sum by pruning the search space. It does this by propagating messages from leaf nodes to the root and only calculates the utility for feasible variable states. We empirically evaluated our algorithm using real distribution network data, showing that it outperformed (in terms of computational time and total size of messages sent) both a centralised approach and the max-sum approach for large networks.

[5] A ring main topology consists of a number of radial networks connected in a ring.

9. REFERENCES

[1] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[2] E. M. Davidson, M. J. Dolan, S. D. J. McArthur, and G. W. Ault. The use of constraint programming for the autonomous management of power flows. In *Proc. of the 15th Intl. Conf. on Intelligent System Applications to Power Systems*, pages 1–7, Curitiba, Brazil, 2009.

[3] Department for Business Enterprise and Regulatory Reform. Active network management (ANM) technology. Technical report, 2008.

[4] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proc. of the 7th Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 639–646, Estoril, Portugal, 2008.

[5] T. Gönen. *Electric power distribution system engineering.* McGraw-Hill New York, 2nd edition, 2007.

[6] M. E. Granada, M. J. Rider, J. R. S. Mantovani, and M. Shahidehpour. Multi-areas optimal reactive power flow. In *Proc. of the Transmission and Distribution Conf. and Exposition*, pages 1–6, Bogota, Colombia, 2008.

[7] N. Hatziargyriou, N. Jenkins, G. Strbac, J. A. Pecas Lopes, J. Ruela, and A. Engler. Microgrids-large scale integration of micro-generation to low voltage grids. *EU contract ENK5-CT-2002-00610, Technical annex*, 2002.

[8] B. H. Kim and R. Baldick. Coarse-grained distributed optimal power flow. *IEEE Transactions on Power Systems*, pages 932–939, 1997.

[9] J. K. Kok, M. J. J. Scheepers, and I. G. Kamphuis. Intelligence in electricity networks for embedding renewables and distributed generation. *Intelligent Infrastructures*, pages 179–209, 2010.

[10] A. Kumar, B. Faltings, and A. Petcu. Distributed constraint optimization with structured resource constraints. In *Proc. of the 8th Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 923–930, Budapest, Hungary, 2009.

[11] P. J. Modi, W. M. Shen, M. Tambe, and M. Yokoo. ADOPT: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[12] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proc. of the 19th Intl. Joint Conf. on Artificial Intelligence*, pages 266–271, Edinburgh, Scotland, UK, 2005.

[13] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. Putting the "smarts" into the smart grid: a grand challenge for artificial intelligence. *Communications of the ACM*, 2011.

[14] D. Roberts. Network management systems for active distribution networks: a feasibility study. *DTI Distributed Generation Programme (Contractor: SP PowerSystems LTD), Contract Number: K/EL/00310/00/00, URN*, 2004.

[15] T. Voice, R. Stranders, A. Rogers, and N. Jennings. A hybrid continuous max-sum algorithm for decentralised coordination. In *Proc. of the 19th European Conf. on Artificial Intelligence*, pages 61–66, Lisbon, Portugal, 2010.

[16] P. Vytelingum, S. D. Ramchurn, T. D. Voice, A. Rogers, and N. R. Jennings. Trading agents for the smart electricity grid. In *Proc. of the 9th Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 897–904, Toronto, Canada, 2010.

[17] B. M. Weedy and B. J. Cory. *Electric Power Systems.* John Wiley & Sons, 4th edition, 2004.

[18] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural computation*, 12(1):1–41, 2000.