Pacific Northwest
NATIONAL LABORATORY

PNNL-35109

Application Deconfliction Characterization and Alternatives Analysis

September 2023

Andrew Reiman
Alexander Anderson
Gary Black
Andrew Fisher
Monish Mukherjee
Shiva Poudel
Tylor Slay
Orestis Vasios
Jim Ogle

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

Application Deconfliction Characterization and Alternatives Analysis

September 2023

Andrew Reiman
Alexander Anderson
Gary Black
Andrew Fisher
Monish Mukherjee
Shiva Poudel
Tylor Slay
Orestis Vasios
Jim Ogle

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

This report provides an overview of the domain space and solution techniques that could be used to create a robust, flexible app deconfliction service. Three approaches are reviewed with summaries of the characteristics, elements, and results from preliminary demonstrations of solution techniques based on each approach: 1) rules and heuristics, 2) cooperation, and 3) optimization.

The strengths and weaknesses of each solution technique were explored through a set of numerical demonstrations on modified IEEE 123 node and 9500 node test feeders. An alternatives analysis of individual deconfliction elements was performed with each solution technique element evaluated against criteria reflecting the dynamic app environment, need to balance app objectives, and scalability issues versus the number of applications, setpoints, and distributed control areas.

It is anticipated that a combined solution for a GridAPPS-D Deconfliction Service can be formulated using a combination of elements from each solution technique. The combined solution would combine 1) device control budgets to reduce the size of the solution space by constraining system setpoints to those will not result in accelerated degradation of physical assets, 2) system operations rules to constrain the solution space by eliminating setpoints that result in violations of system limits or operational best practices, 3) contextual status signals shared with or among apps such that they could update their desired setpoints based on the evolving context, 4) a mediator that incentivizes apps to come to a cooperative solution, and 5) Setpoint-informed optimization as a fallback mechanism if a cooperative solution cannot be agreed upon by applications.

Acknowledgments

This work was supported by the U.S. Department of Energy (DOE), Office of Electricity, Advanced Grid Research Program.

Acronyms and Abbreviations

ADMS	Advanced Distribution Management System
CIM	Common Information Model
CVR	Conservation Voltage Reduction
DER	Distributed Energy Resource
DSO	Distribution System Operator
EIA	Energy Information Administration
JSON	JavaScript Object Notation
MAS	Multi-Agent System
MCDM	Multi-Criteria Decision Making
MILP	Mixed Integer Linear Program
SMARTER	Simple Multi-Attribute Rating Technique Exploiting Ranks
SoC	State of Charge
WAPA	Western Area Power Administration
XML	eXtensible Markup Language

Contents

1.0 Introduction

This report summarizes and synthesizes approaches from different domains for solving the "app deconfliction" problem. that arises when different applications – or apps – in an advanced distribution management system (ADMS) or similar app-hosting platform attempt to control the same devices or actuators. In advanced distribution operations, the introduction of an open data-integration platform promises two advantages over vertically integrated platforms: 1) the open platform increases the total decision space available to apps by removing hidden restrictions on app functionality, operating conditions, and access to devices that effectively create orthogonal decision spaces for each app in order to avoid conflict; and 2) the open platform enables implementers to choose a set of best-of-breed apps rather than those paired with a vertically integrated solution. In this way, the open platform will tend to lead to device-setpoint conflicts as independently developed apps attempt to achieve their best outcomes by controlling as many devices as they can.

Prior work identified a formal deconfliction problem [1] and three categories of techniques – or solution domains – that can be used to solve multi-criteria, multi-stakeholder, and/or multi-objective problems like the deconfliction problem; namely, 1) rules & heuristics, 2) cooperation or negotiation, and 3) optimization. These three solution domains were investigated independently under the premise that solutions could be decomposed into elements and that elements from any or all domains can be recombined to describe a wide range of multi-domain strategies. In Section 2, a summary characterization is provided for each solution domain. In Section 3 a demonstration testbed is described. In Section 4, at least one example solution technique from each solution domain is presented. The parameters for each example were selected to highlight unique aspects of the corresponding domain and technique; therefore, example scenarios are technique-specific, and results cannot be compared side-by-side. In Section 5, an alternatives analysis examines elements of each solution domain for suitability in the GridAPPS-D environment. GridAPPS-D is an open-source platform that supports the development and maturation of modular and distributed apps for advanced grid operations [2]. A technique that combines elements from each solution domain is described. This combined-technique description will inform design and implementation of an initial GridAPPS-D deconfliction service.

2.0 Background

This section summarizes prior work that characterized three domains of solutions to multi-criteria, multi-stakeholder, and/or multi-objective problems. The solution domains were described based on the premise that individual solutions can be decomposed into elements, which can be recombined to describe a wide range of possible solutions to the deconfliction problem. Section 2.1 describes using rules and heuristics to solve the deconfliction problem. Section 2.2 describes using cooperative multi-agent frameworks to solve the deconfliction problem. Section 2.3 describes using optimization to solve the deconfliction problem.

2.1 Rules & Heuristics Characterization

The rules-based deconfliction methodology is the first of three alternative numerical solution techniques that can be used within the deconflictor. The approach applies a set of asset-based heuristics to reduce the size of the solution domain and then select the combination of device setpoints that results in an optimal tradeoff between technical, economic, environmental, and social decision criteria selected by the distribution system operator (DSO).

Selection of the decision criteria used within the deconflictor is based on the operational priorities of the DSO and can incorporate objective functions shared by apps to incentivize cooperation. The decision criteria are ranked by the DSO from most important to least important for each set of operating conditions and then weighted based on their relative importance. The set of criteria and their importance ranking may change based on evolving grid conditions as the system degrades from “normal” to “alert” to “emergency” operations. Thus, during “normal” operations, the DSO may value operating profit and carbon emissions as most important, but during a storm, restoration times and customers served may be the most important criteria [3].

The set of device setpoints that result in an optimal tradeoff between the selected decision criteria can be located through a discrete optimization approach or discrete multi-criteria decision-making (MCDM) techniques. In [4], [5], the simple multi-attribute rating technique (SMARTER) [6] was used to select discrete alternatives formulated as combinations of discretized device setpoints. The degree to which each setpoint combination satisfied the decision criteria was evaluated through snapshot power flows calculated for the given timestep with all devices set to the selected setpoint or control mode. This approach is useful for both devices with discrete controls (such as voltage regulator taps and capacitor banks) and new inverter-based resources controlled through specification of control modes via the MESA-DER protocol [7]. Implementation of discrete inverter control modes (such “volt-watt” and “power-factor-correction”) as discrete alternatives within an MCDM framework is much simpler than within a multi-objective optimization scheme.

To avoid multiplicative growth of number of alternatives to be evaluated, a distributed approach was used, in which devices were grouped into local control areas based on the network topology. Three approaches were compared in [5] to decompose the deconfliction: 1) by grouping all devices within each topological area, 2) by grouping all devices connected to each phase, and 3) by considering each device separately in a fully decentralized manner. The fully decentralized method was shown to produce near-optimal results with significantly reduced computation times.

To further reduce the size of the solution space and number of alternatives to be considered, the rules-based deconfliction methodology applies a set of heuristic asset rules derived from the temporal reservoir drainage water budgets used for over fifty years in dispatch of hydroelectric power plants [8], [9]. Device controls budgets are established for each asset by the DSO and specify the maximum number of duty cycles and actions that impact asset lifespan that may be performed over a given time interval. Examples of controls budgets include battery charge/discharge cycles per day, regulator tap changes per hour, and transformer winding overheating minutes per year. A full list of 30 asset-centric technical, economic, environmental, and social rules are given in [4]. The deconflictor may use these asset rules to evaluate whether any setpoints violate any of the selected rules and eliminate those setpoints from the solution space prior to solving snapshot power flows and calculating criteria scores.

2.2 Cooperation Summary Characterization

Cooperative decision-making has been a commonly used approach for conflict resolution in multi-agent system (MAS) [10] for various engineering apps, including distributed artificial intelligence, building energy and comfort management, and multi-microgrid coordination. Stemming from the general theory of cooperation, this approach focuses on a cooperative decision-making framework for conflict resolution among distribution system apps, thereby enabling independently developed apps to operate together in a compatible and constructive way.

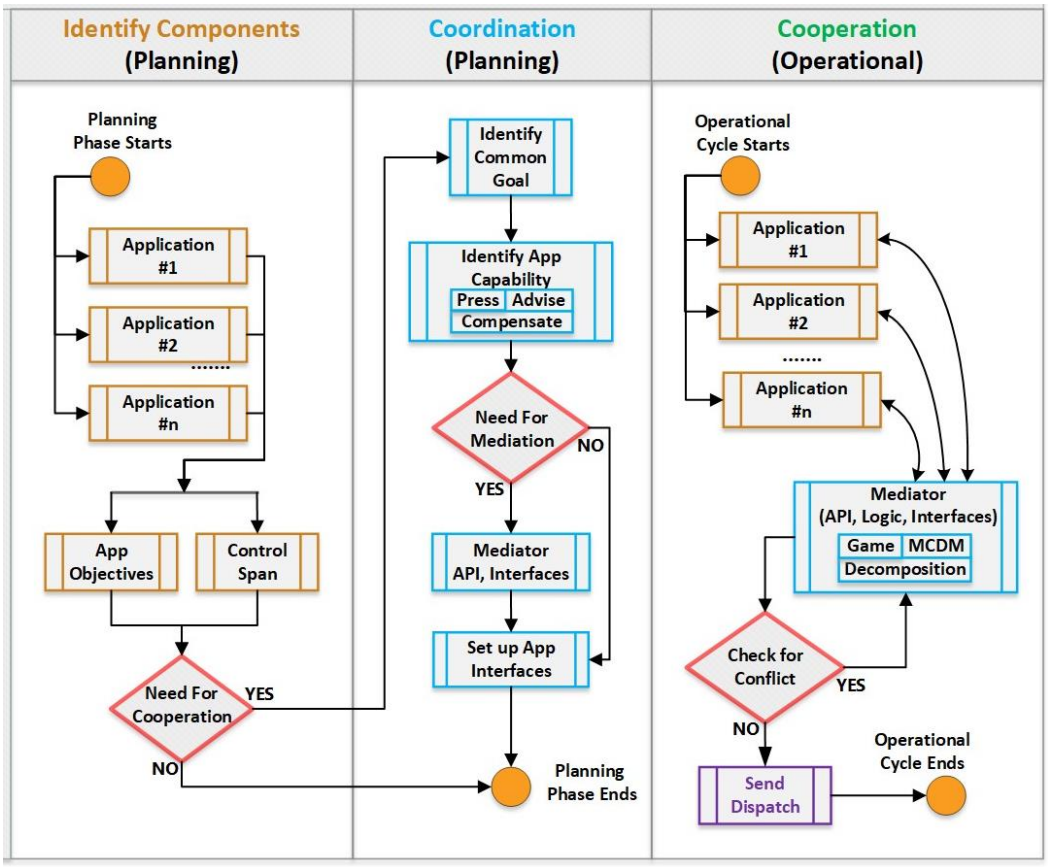


Figure 1: Process flow for enabling cooperation among distribution apps [25]

From the operations perspective of an electric power distribution system, conflicts may arise in modular platforms when multiple apps with different objectives share controllable resources. These conflicts may create the following operational issues: arbitrary response of devices, oscillating system behavior, or apps failing to achieve their respective objectives. Existing literature presents a set of conceptual process elements for enabling cooperation in MAS [10], [11]. Cooperative decision-making for conflict resolution among distribution system apps can be achieved through systematically implementing a cooperative process. Figure 1 presents the flow of the processes involved in enabling cooperative behavior among apps. To begin the process of cooperation, the primary components driving the need for cooperation need to be identified. From the DSO perspective, these include the objectives of the apps, their span of control, and any overlap in their goals. The next stage of achieving cooperation is enabling coordination among the agents, which aims at the joint determination of common goals. From the perspective of the apps, such goals can be the ability to trade-off objectives and control resources in a shared fashion. In order to enable apps to achieve deconfliction cooperatively, potential mediation techniques for conflict resolution need to be identified.

Mediation for conflict resolution can be realized through numerical techniques which can be broadly categorized into a combination of *press*-, *compensate*-, *advise*-, and *ignore*-based actions. Cooperation can be enabled by implementing the conflict resolution techniques to achieve the identified common goals. This can be achieved through systematically determined contextual signals that can influence the behavior of the apps through a combination of the above-mentioned actions towards archiving a cooperative trade-off among all participants.

There exists a wide range of cooperative solutions available to influence the behavior of apps through a combination of contextual signals (*press*, *compensate*, *advise*, and *ignore*) towards a desirable compromise for all participating actors. Some of the primary solution techniques include *game theory*, *multi criteria decision making* (MCDM), and *decomposition-based methods*. Game theory is a key mathematical tool for solving decision-making problems involving multiple entities or players. Game theory techniques can be broadly categorized into 1) non-cooperative games that focus on strategic decision-making problems among greedy players with conflicting objectives or payoffs over the strategy space and 2) cooperative games that focus on stable coalition among players, in the presence of some binding agreements, that can improve their overall utility. MCDM techniques focus on subjective evaluation of criteria based on combining conflicting qualitative and quantitative goals into a single weighted objective. Decomposition techniques present an approach to transforming a multi-objective decision-making problem into simpler subproblems through a weighted sum of objective functions that would enable individual apps to solve their objectives separately by adjusting their decision variables based on information received from other participants toward achieving a global objective.

2.3 Optimization Summary Characterization

Optimization strategies play a key role in the operation and control of the power grid; we even expect that many (or even all) of the individual apps of interest will implement an optimization problem internally to aid them in deciding their setpoint requests. We started our effort with an exhaustive mapping of the sources and types of information that the deconflictor can utilize to build and solve an optimization problem to determine deconflicted setpoint commands [12].

The first potential source of information to the deconflictor is, of course, apps themselves. They can provide mathematical representations of their internal models and/or goals, thus allowing the deconflictor to get a better understanding of their operation and reach a more optimal conflict resolution. This necessitates a pre-agreed communications framework between apps and the deconflictor to ensure efficient exchange of information that is comprehensible by both sides of the exchange. It is worth noting here that the only strict requirement placed on individual apps that wish to engage with our deconfliction framework is to provide setpoint requests for the grid devices they want to control. All additional types of information are optional.

The deconflictor can obtain supplementary information from enabling systems, as well as knowledge of system constraints. Enabling systems are systems other than apps that provide the deconflictor with awareness of the past, present, and future grid conditions. These include databases that store historic information about the grid, measuring devices that supply information on currents and voltages at different parts of the power system, and load and temperature forecasts. System constraints are the requirements placed on the power grid by the laws of physics and society. In this category we can find power flow equations, limits on the voltage supplied to customers, and renewable energy integration mandates. Information from these non-app sources can also be mathematically modeled and passed to the deconflictor.

Our optimization-based deconfliction methods should at a minimum be able to calculate deconflicted setpoints when individual apps provide only the required setpoint requests. We have already described how the deconflictor utilizes setpoint requests to detect conflict [1]. Once this step is finished, an optimization-based deconflictor can proceed to conflict resolution for each device setpoint control variable over which a conflicted was detected by minimizing the distance between that variable and all app-provided setpoint requests. Equations obtained from enabling systems as well as system constraints can be used at this stage as constraints to the minimization problem.

On the other hand, the deconflictor can also obtain the full internal optimization problems from conflicting apps, aggregate them, augment them with potential information added by enabling systems and/or system constraints, and build a complete optimization problem that describes system operation. By solving this problem, deconflicted setpoint commands for the grid can once more be obtained.

The proposed optimization-based deconflictor can handle both the minimum information scenario of knowing only setpoint requests and the maximum information scenario of full visibility into app internal optimization problems. For intermediate points on the information spectrum two definitions are needed. Utility functions mathematically represent the objectives and preferences of the corresponding app. App-provided constraint functions describe the views of the corresponding app on what constitutes permissible system operation. Apps can choose how detailed their provided functions are up to and including using their actual internal objective functions and constraints. In case of conflict apps can elect to provide at least one utility function and/or at least one app-provided constraint function. The deconflictor can then use this information for the formation of a deconflictor-level optimization problem whose solution can provide deconflicted setpoint commands.

3.0 Simulation Testbed

In this section we introduce a simulation testbed featuring a set of competing apps utilized in the evaluation of deconfliction methods.

3.1 GridAPPS-D Deconfliction Software Framework

A prototype deconfliction software framework serving as a simulation testbed has been implemented in Python and consists of several interoperating components: 1) Deconfliction Simulator; 2) Competing Apps; 3) Deconfliction Methods; and 4) Deconfliction Pipeline. The Deconfliction Pipeline coordinates the other components communicating with them over the GridAPPS-D message bus (Deconfliction Simulator and Competing Apps) or Python module import and class invocation (Deconfliction Methods). Figure 16: Deconfliction software framework with interoperating components. The arrows represent GridAPPS-D messaging.

Figure 17: Battery and regulator taps requested by each app, compared to deconfliction solution decomposed by switch-delimited topological area.Figure 18: Deconfliction software framework with interoperating components. The arrows represent GridAPPS-D messaging. shows the interoperation of all the software framework components including the data passed between them over the message bus. Time-series-based load and solar profiles along with device status data (e.g., battery SoC) are generated by the Deconfliction Simulator and published on the message bus. One or more Competing Apps running concurrently consume profile and device data and produce new device setpoint requests published back to the message bus. The Deconfliction Pipeline maintains an evergreen Conflict Matrix data structure based on setpoint requests from the set of running Competing Apps and invokes the Deconfliction Method to produce a Resolution Vector data structure. The Resolution Vector is then translated into device setpoints also put on the message bus. Finally, the Deconfliction Simulator processes the deconflicted setpoints dispatching them to devices and for input in generating subsequent profile and device status data completing the roundtrip workflow.

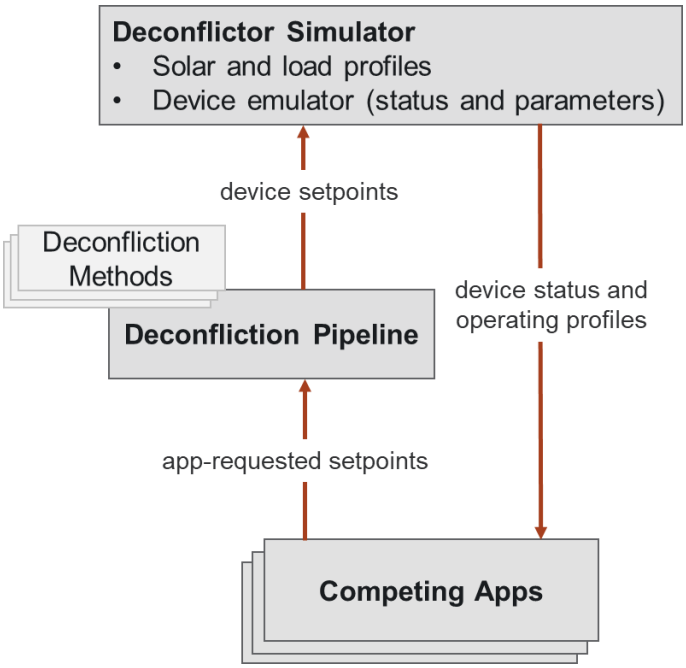


Figure 16: Deconfliction software framework with interoperating components. The arrows represent GridAPPS-D messaging.

Figure 17: Battery and regulator taps requested by each app, compared to deconfliction solution decomposed by switch-delimited topological area.Figure 18: Deconfliction software framework with interoperating components. The arrows represent GridAPPS-D messaging.

The two primary tasks of the Deconfliction Pipeline are maintaining the Conflict Matrix, as well as directing and supporting the Deconfliction Method in creating Resolution Vectors from the Conflict Matrix snapshots. In the initial prototype implementation, the Deconfliction Pipeline performs all GridAPPS-D messaging on behalf of the Deconfliction Method. The Conflict Matrix stores the most recently requested setpoints for all running competing apps over all devices. The Deconfliction Pipeline must account for the asynchronous nature of competing apps, which not only can request setpoints on an everchanging time scale, but also stop completely and restart at a later time. Due to this unpredictable nature of competing apps requesting setpoints, the Deconfliction Pipeline design dictates a Resolution Vector be produced immediately for every setpoints request. In the prototype implementation, this can result in several different setpoints being dispatched to the same device in quick succession when multiple competing apps are requesting setpoints based on the same time-series profile data. Future enhancements will curtail these dispatches when they would compromise device reliability/longevity such as with multiple quick or large regulator tap position changes.

The Deconfliction Method is a dynamically configurable or pluggable module for the Deconfliction Pipeline implemented via the standard Python import capability in the prototype. The responsibility of the Deconfliction Method is generating a Resolution Vector structure from the Conflict Matrix provided by the calling Deconfliction Pipeline. A Resolution Vector consists of a single setpoint value, deconflicted based on the underlying technique that has been implemented within the method, for each device appearing in the Conflict Matrix. Deconfliction Methods are implemented as a Python class with a class method being called to perform deconfliction and produce a Resolution Vector. Basic sample classes have been created that implement exclusivity with a single competing app controlling setpoints and simple compromise taking the mean of device setpoint requests between a specified set of competing apps to produce a deconflicted setpoint. The deconfliction software framework was implemented as a testbed for the investigation, development, testing, and refinement of the deconfliction techniques and methodologies described in this report. The pluggable Deconfliction Method class provides a simple standardized means for deconfliction technique integration while a testbed wrapper script for invoking all software framework components from a single point also supports deconfliction technique development.

3.2 Competing Apps

We have developed three competing apps with the primary objective of illustrating operational conflicts. These apps serve purely demonstrative purposes and have intentionally constrained functionalities (as compared to commercial ADMS apps). It is essential to note that they do not reflect the overarching goals or the level of sophistication typically associated with real-time operational systems.

Competing apps are developed in two distinct versions. The first version adopts a workflow-centric design approach, crafting apps through condition-based decision-making as described in [1]. In contrast, the second version employs an optimization-oriented formulation. For simplicity, we have chosen linear objectives for each of the competing apps. The distribution network is modeled using linear constraints with binary decision variables. As a result, the optimization problem for the competing apps is structured as mixed-integer linear programs [13]. A short description of each competing app is given below:

1) *Resilience*: The primary goal of the Resilience App is to optimize the system's instantaneous reserve capacity. It accomplishes this by both charging the batteries and maintaining the highest state of charge (SoC) to uphold adequate reserves. Thus, the Resilience App charges batteries regardless of the power source, whether it's from the grid or renewable energy.

2) *Decarbonization*: The objective of the Decarbonization App is to reduce reliance on grid power, which is assumed to be predominantly generated from fossil fuels. It achieves this by minimizing the need for importing energy from the grid. Simultaneously, the app ensures that surplus solar power within the feeder is harnessed efficiently, i.e., supplying local loads and charging batteries.

3) *Conservation Voltage Reduction (CVR)*: The CVR App reduces energy consumption by decreasing the voltage supplied to end-users while ensuring performance levels remain

within acceptable bounds. This app can yield energy savings by optimizing voltage levels, directly leading to reduced electricity procurement expenses, particularly during periods of peak demand. In our work, the CVR App is designed by the utilization of voltage regulators and battery dispatches aimed at reducing voltage levels.

3.3 Simulation Setup

Load and solar profiles are generated at 15-minute intervals over a 24-hour period for the demonstration. Device status information, specifically battery SoC values, are provided along with the profile data. This set of profile and device status information is taken by any running competing apps to produce new device setpoint requests. Two different test cases are used in developing and testing the performance of competing apps. A brief description of each test case is provided below.

- *Test Case I*: The 9500-node test feeder [14] is used in the development of workflow-based competing apps where two batteries are considered as the controllable devices. The Common Information Model (CIM) representation of the test feeder model is maintained within a GridAPPS-D Blazegraph database and is publicly available in extensible markup language (XML) format[1].

- *Test Case II*: The original IEEE 123-bus is modified to include batteries and solar PVs. This modified test feeder is leveraged by the optimization-based competing apps where the controllable devices are batteries and regulators. The CIM representation of this test feeder model[2] should be uploaded into the GridAPPS-D Blazegraph database before starting a simulation.

[1] https://github.com/GRIDAPPSD/Powergrid-Models/tree/develop/platform/cimxml
[2] https://github.com/GRIDAPPSD/app-deconfliction/tree/main/competing-apps/sim-starter

4.0 Illustrative Examples of Solution Techniques

This section summarizes a specific solution from each of the three domains characterized in Section 2. The testbed described in Section 3 was used to demonstrate the specific solutions. Results from each solution demonstrate strengths and weaknesses of each domain; however, each solution was designed according to and evaluated based on domain-specific criteria and results cannot be compared directly.

4.1 Rules & Heuristics

The rules-based deconfliction methodology was formulated for the three competing apps described on the modified IEEE 123 bus test case with additional DER, described in Test Case II above. The deconfliction methodology was implemented as a microservice that received only the Conflict Matrix of competing setpoints at each timestep and returned a Resolution Vector of deconflicted setpoints after completion of the deconfliction solution.

4.1.1 Formulation of Snapshot Power Flows

To create predictive estimates of system outcomes for each of the combinations of setpoint alternatives, the deconfliction system implemented a bridge to OpenDSS through the OpenDSSDirect.py package[1]. The setpoints received through Conflict Matrix were separated into a minimum and maximum for each device, forming the bounds on the available solution space. The solution space was subsequently discretized into individual alternatives. For devices with discrete controls (e.g. regulator taps), each intermediate discrete setpoint between the minimum and maximum available as an alternative. For devices with continuous ranges (e.g. inverters), the range was divided into equal intervals with solution resolution adjustable by the user. Setpoints that violated any of the device controls budgets described below were eliminated prior to solving the snapshot power flows. Setpoints were then grouped into distributed control areas using a configuration file that listed the controllable devices in each area, as determined using the GridAPPS-D Topology Processor Service [15].

Each combination of deice setpoints was passed to OpenDSS by modifying the device setpoints in the active circuit object and running a snapshot power flow. The solution results were obtained by iterating through each device class to obtain key measurement data needed to calculate the decision criteria and to determine whether any system-level rules were violated by the given setpoint combination. The decision criteria values were calculated and assembled into a matrix and then normalized into non-dimensional utility functions for use within the SMARTER multi-criteria decision-making framework.

4.1.2 Formulation of System and Asset Rules

Four system and asset health rules were selected for implementation through constraints and controls budgets. The first was restricting the voltage of all nodes to remain between 0.95 pu and 1.10 pu. The second was limiting the total apparent power flow through the source bus (equivalent to a substation transformer) to 5000 kVA. Two rules were formulated for voltage regulators based

[1] https://github.com/dss-extensions/OpenDSSDirect.py

on vendor nameplate data [16], where winding current was limited to 668 A. Additionally, a controls budget of 6 tap changes per hour (approximately equal to one million tap changes over a twenty-year lifespan) was set for all regulators. The formulation included a fallback "do-nothing" solution to return the current device settings as the Resolution Vector if all requested setpoints from the apps and all combinations of setpoints resulted in violations of the system and asset rules.

4.1.3 Formulation of Decision Criteria

For the deconfliction demonstration, three system-level metrics (technical, economic, and environmental) and the shared objective function of one app were chosen. The selected technical decision criterion was the total real power loss in the network, which was obtained from the snapshot power flow results. The utility function for this criterion was assumed to decrease linearly with network losses.

The selected economic criterion was calculated as the revenue from serving retail distribution loads and cost of purchases from the transmission system, net metering agreements, and fuel cost of utility-owned distributed generation:

$$Profit = c_r P_{load} - c_t P_{trans} - c_n P_{net_{mtr}} - c_{dg} P_{dg}$$

The values of the coefficients were derived from publicly available utility documents for the state of Colorado, as summarized in Table 1. The utility function for this criterion was assumed to increase linearly with profit.

Table 1: System-level Coefficient Values

Coefficient	Description	Value	Reference
c_t	Cost of energy purchased from transmission grid	$0.02817/kWh	[17]
c_r	Retail distribution tariff rate	$0.1090/kWh	[18]
c_n	Net metering tariff rate	$0.01786/kWh	[19]
c_{lng}	Cost of natural gas DER	$0.25/kWh	
c_{dies}	Cost of diesel DER	0.34/kWh	
e_{trans}	Emissions from energy purchased from transmission grid	1.205 lb/kWh	[18]
e_{lng}	Emissions of natural gas DER	0.97 lb/kWh	[20]
e_{dies}	Emissions of diesel DER	2.44 lb/kWh	[20]

The selected environmental decision criterion was total CO2 emissions from the feeder, which was expressed as the sum of emissions from power purchased from the bulk transmission system, emissions from diesel DERs, and emissions from natural gas-fired DERs:

$$Emissions = e_{trans}P_{sub} + e_{dies}P_{dies} + e_{lng}P_{lng}$$

The values of the coefficients were derived from EIA datasets for Colorado and common fuel sources, as summarized in Table 2. The utility function for this criterion was assumed to decrease linearly with emissions.

The last decision criterion was the objective function of the Resilience App, which was shared by the developers and is included to incentivize cooperation among the apps. The utility function for this criterion was assumed to increase linearly with the total battery charging power.

4.1.4 Comparison of Decomposition Methods

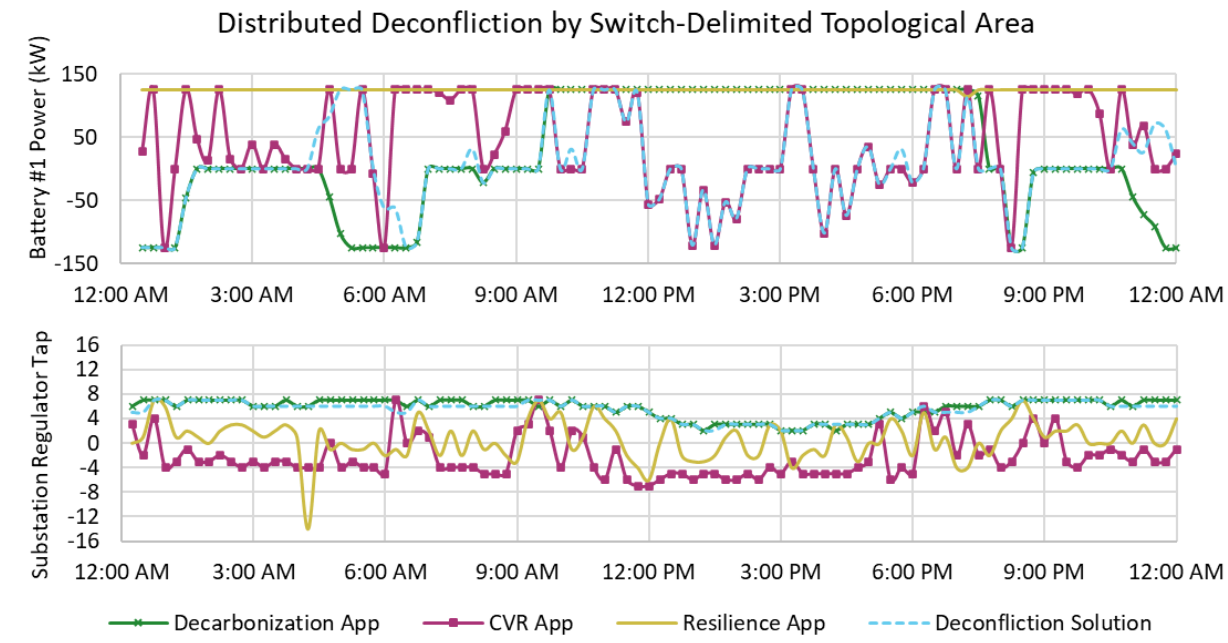
The rules-based deconfliction service was run in the testbed for a 24-hour simulation period using a decision criterion preference ranking (from most important to least important) that profit > losses > emissions > resilience. The distributed solution applied three different methods for decomposing the deconfliction problem. A distributed deconfliction solver was established for each control area and sequentially solved the deconfliction problem for that area using the results of the lower downstream deconfliction solution.

The first method divided the feeder into switch-delimited topological areas with a substation and five downstream distributed control areas that represent the portion of the feeder between an upstream and downstream switch(es). The second method grouped devices by the phase on which they are connected, such that the deconfliction agent formed setpoint alternatives for devices on each phase (A, B, C) and then for 3-phase devices. The third method was fully decentralized and deconflicted each device individually.

Table 2: Cumulative 24-hr System Outcomes for each Decomposition Method

Method	Average Solution Time (s)	Total Profit ($)	Total Losses (kWh)	Total Emissions (ton CO2)
By Switch Area	0.46	2148	631	-531
By Phase	1.22	2135	621	-412
Fully Decentralized	0.14	2147	635	-367

The results from the decomposition methods are largely similar, as can be seen from Table 2 and Figures 3 and 4. All of the apps are able to receive their preferred setpoints over certain periods of the day. During the night, Decarbonization App setpoints are selected when importing from the bulk grid. During the daytime, setpoints from the CVR App are often chosen to maximize profit. Negative emission values are caused by net backfeed from photovoltaic DERs.



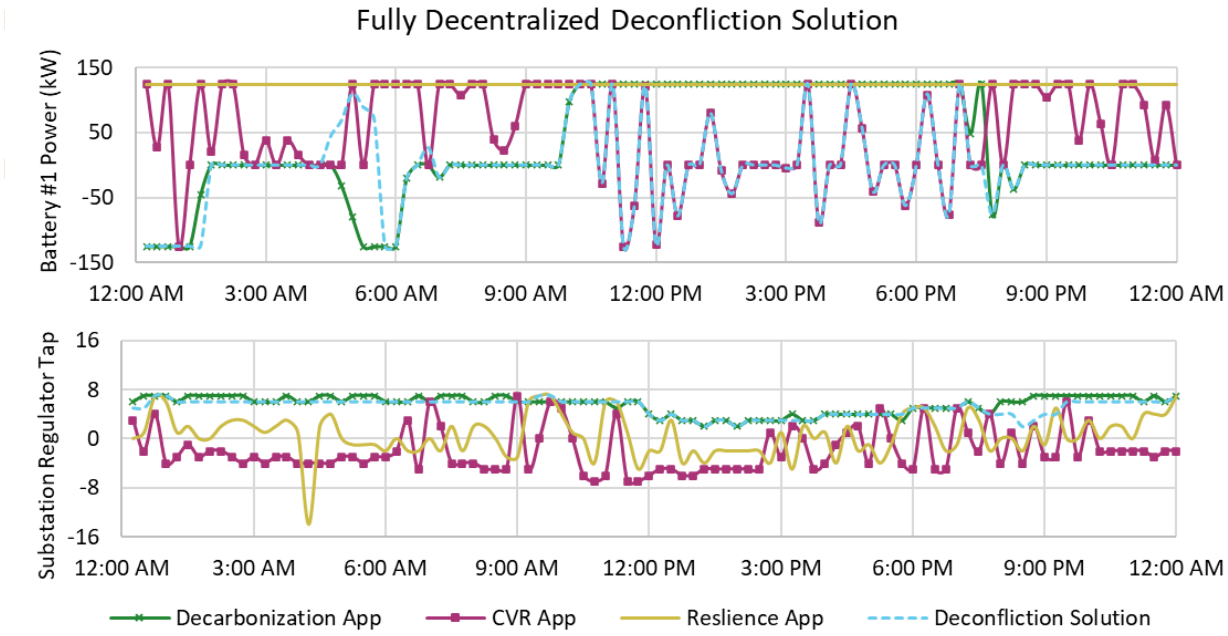


Figure 22: Battery and regulator taps requested by each app, compared to deconfliction solution considering each device separately in a fully decentralized manner.

Figure 23: Simulation conditions along with for app-preferences (left) and battery-dispatch with naïve and cooperative apps with mediation (right). The shaded portion indicates the operational status during outage interval.Figure 24: Battery and regulator taps requested by each app, compared to deconfliction solution considering each device separately in a fully decentralized manner.

4.2 Cooperation

The simulation scenario outlined in Section 3.3 for *Test Case I* was used with the following modifications:

- Only Decarbonization and Resilience Apps were compared to simplify the definition of conflict between apps. Alongside normal operations, an outage event was simulated between 18:00-21:00 PM where the grid supply was considered to be interrupted and the Resilience App operates exclusively to dispatch the batteries to maximize the loads that can be served.

- The two controllable batteries, used for the test case, were assumed to start with their minimum SoC levels.

- The objectives of the Decarbonization and Resilience Apps were modified to include a burden constraint on the mediators' advice based on the following conditions:

    1) If the compensate signal was greater than the burden; the advice would be the accepted setpoints by the Apps.

    2) Otherwise, the apps would update their setpoints to minimize the burden based on the press signal received.

$$\text{burden} = \frac{dist\left(SP_{App1}^{D1}, SP_{advice}^{D1}\right)}{dist\left(SP_{App1}^{D1}, \max\right)}$$

The mediator only intercedes when there is conflict with another app. When a conflict is detected it first attempts to advice the apps with a centroid setpoint between the conflicting apps. If the apps do not accept the advice, then a press or compensate signal is used to drive cooperation between conflicting apps.

When the press signal is used it is sent to each app along with the advice. When new setpoints are received the penalty is calculated and applied to the apps setpoints. If conflict is reduced to acceptable margins the new setpoints are sent to the DER. If conflict is not resolved the press signal is increased and a new advice is calculated based on the mediated setpoints with the previous penalty.

$$advice = centroid(app1, \dots, appN)$$

$$penalty = press * dist(setpoint, advice)$$

$$setpoint_{mediated} = (1 - penalty) * setpoint$$

$$conflict = average(centroid(app1, \dots, appN))$$

Intelligent apps will respond with modified setpoints based on the advice and penalty / compensation signal they have received driving them closer to the advice. If an app does not or cannot respond it will be either not receive the compensation or experience the full penalty. This ensures flexible apps still experience some percentage of their objective without completely compromising with an app that refuses to participate.

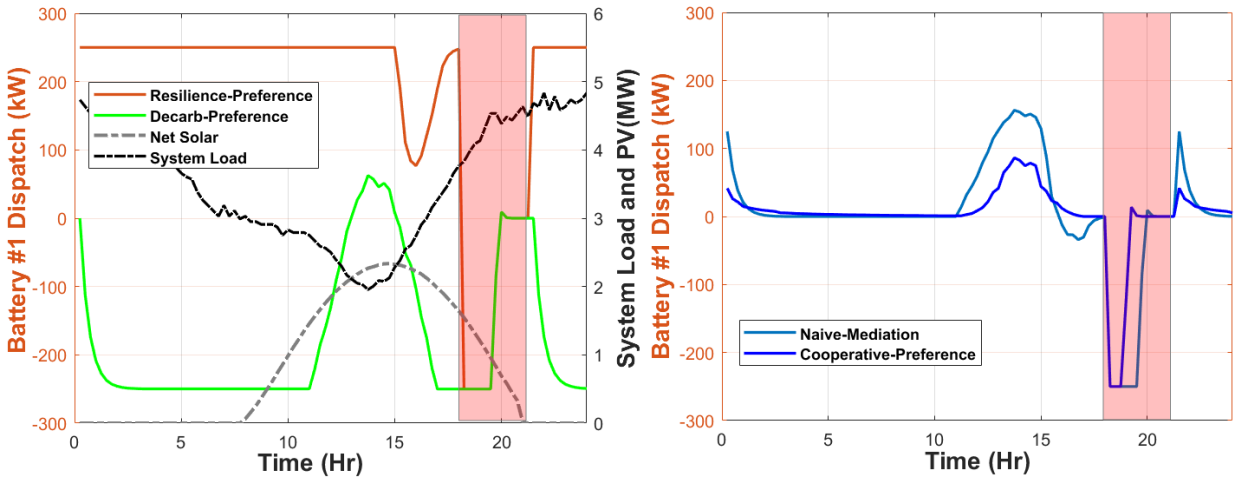


Figure 25: Simulation conditions along with for app-preferences (left) and battery-dispatch with naïve and cooperative apps with mediation (right). The shaded portion indicates the operational status during outage interval.

Figure 26: Simulation results for cooperative apps with continuous- (left) and blocking-based (right) solutions.Figure 27: Simulation conditions along with for app-preferences (left) and battery-dispatch with naïve and cooperative apps with mediation (right). The shaded portion indicates the operational status during outage interval.

Figure 5 (left) demonstrates system conditions for the chosen simulation scenario along the dispatch preferences for the Decarbonization and Resilience Apps if operated exclusively. As outlined in Section 3.2, the objective for decarbonization is to reduce overall import of electricity and capture excess solar. This objective can be summarized by minimizing the overall distance between solar and demand, whereas the objective for Resilience can be summarized by maximizing the power delivered:

$$Performace^{App} = \frac{App^{Cooperative} - App^{base}}{App^{Exclusive} - App^{base}}$$

$$Decarbonize^{objective} = -\sum dist\,(solar, load + batteries)$$

$$Resilience^{objective} = \sum load - solar + batteries,$$

where $base$ denotes the base app performance determined by excluding batteries throughout the entire day, $exclusive$ refers to the exclusive app performance determined by only allowing control of batteries by one application throughout the entire day, and $cooperative$ describes the cooperative app performance determined by comparing the mediated control between cooperating Apps with their base and exclusive operation.

Table 3 and Table 4 highlights the performance of the Decarbonization and Resilience App respectively. There is an additional indicator of load lost during fault period and excess solar to highlight the effects of each App's objective. There are five scenarios:

- Exclusive without dispatch to create a baseline performance of the application objective with no participation from batteries.
- Exclusive with dispatch to determine what the app would consider ideal performance.
- Naïve cooperation where neither app participates in deconfliction through mediation.
- Cooperative Decarbonization and naïve Resilience to highlight effect of unfair participation.
- Cooperative Decarbonization and Resilience to demonstrate benefits of both applications participating to the best of their ability.

Table 3: Decarbonization App Performance

Description	Performance (%)	Excess Solar (W)	Load Lost (W)
Cooperative Decarbonization & Resilience	0.51	0.00	47597
Only Cooperative Decarbonization	0.11	0.00	44576
Exclusive Decarbonization	1.00	2.76	51895
Exclusive Decarbonization (Without Dispatch)	0.00	1693.76	51895
Naïve Decarbonization & Resilience	-0.01	0.00	44024

Table 4: Resilience App Performance

Description	Performance (%)	Excess Solar (Wh)	Load Lost (W)
Cooperative Resilience & Decarbonization	0.35	0.00	47597
Only Cooperative Decarbonization	0.54	0.00	44576
Exclusive Resilience	1.00	1693	42901

Exclusive Resilience (Without Dispatch)	0.00	1693	51895
Naïve Resilience & Decarbonization	0.58	0.00	44024

Table 5 compares the average performance between each of the apps for each of the operating scenarios. The three scenarios are: two intelligent apps, intelligent Decarbonization with a naïve Resilience, and both naïve apps. A direct average of performance was used assuming each application has the same weighting. The two intelligent apps scenario demonstrates the highest performance. A perfect cooperation solution would yield an equal split in performance; however, each application chooses its own flexibility through its burden calculation. The dispatch setpoints of Battery#1 for the scenarios where both apps are naïve and both apps are intelligent respectively is shown in Figure 5 (right).

Table 5: Cooperation Performance Comparison

Description	Decarbonization	Resilience	Average	Excess Solar (W)	Load Lost (W)
Cooperative Resilience & Decarbonization Apps	0.51	0.35	0.43	0.00	47597
Only Cooperative Decarbonization App	0.11	0.54	0.33	0.00	44576
Naïve Resilience & Decarbonization Apps	-0.01	0.58	0.29	0.00	44024

In order to characterize the behavior of cooperative solution with respect to the time-synchronization, an additional set of simulations were evaluated under the following scenarios:

- ***Continuous – Cooperative:*** This scenario assumes that the interim setpoints of the apps are continuously sent for dispatch each time the apps update their preferred setpoints based on the cooperative signal.

- ***Blocking – Cooperative:*** For this case the apps are permitted to iteratively update their preferred setpoints based on the cooperative signal before a solution is sent for dispatch.

Figure 6 shows the simulation results for the continuous- and the blocking-based cooperation cases respectively. The results indicate the cooperative solution both the cases achieve almost similar trends for the deconflicted-setpoints. Although the continuous case alleviates the synchronization requirements for the cooperative solution, there is a higher volatility in the dispatched setpoints. This can be attributed to the cooperative behavior of apps as they iteratively update their preferred setpoints towards reaching a trade-off among all participants and their interim preferred setpoints are continuously dispatched by the deconflcitor.

4.3 Optimization

The optimization-based approach to deconfliction was demonstrated on the modified IEEE 123-bus feeder described in Section 3 using the Resilience App and Decarbonization App.

4.3.1 Problem Formulation

Both competing apps implement mixed integer linear programming (MILP) problems internally; for simplicity, the same type of problem was chosen for the optimization-based Deconflictor. In order to build a MILP problem for deconfliction purposes, one objective function and one or more constraints are needed.

The Deconflictor objective function is built by combining the utility functions of participating apps with appropriate weights. Specifically, the Resilience App maximizes the sum of the SoCs of the

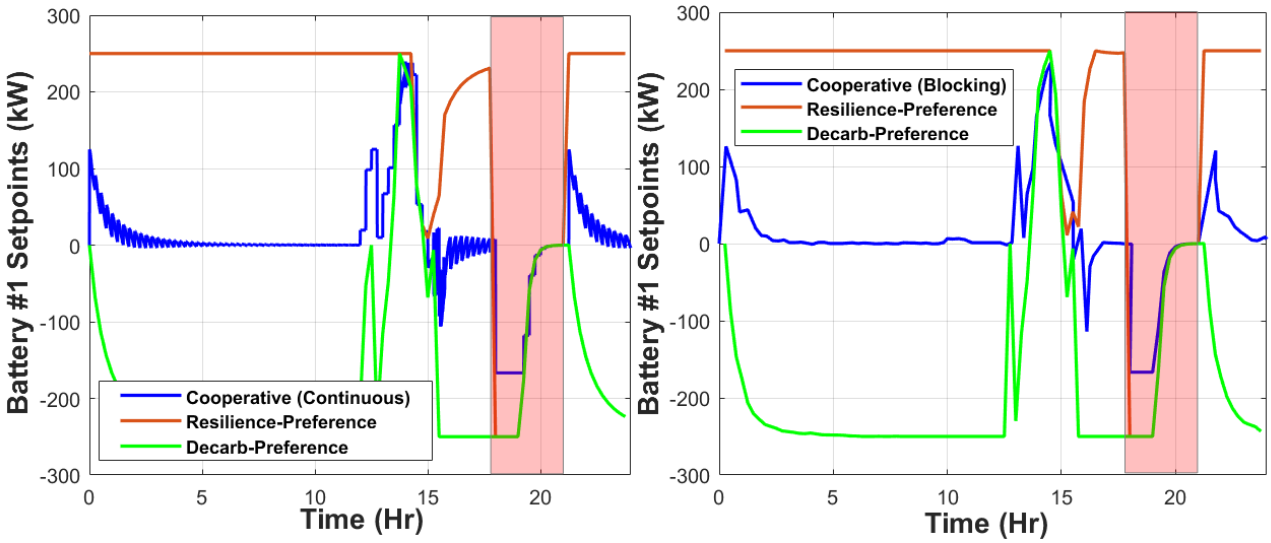


Figure 28: Simulation results for cooperative apps with continuous- (left) and blocking-based (right) solutions.

Illustra Figure 29: Optimization-based deconfliction results for battery power (left) and SoC (right) with the weight, α, equal to 0.0, 0.5 and 1.0.Figure 30: Simulation results for cooperative apps with continuous- (left) and blocking-based (right) solutions.

batteries installed in the feeder. By dividing the internal app objective function by the total number of batteries we obtain a utility function that takes values in the $[0, 1]$ interval. Moreover, the Decarbonization App minimizes the absolute value of the power flowing on the interconnection between the simulated feeder and the external sub-transmission network. Thus, we can obtain an appropriately normalized utility function by dividing the absolute value of the power flow by the kVA limit of the interconnection. Before summing the two utility functions we multiply the Decarbonization App utility function by an operator-defined weight α, and the Resilience App utility function by $(1 - \alpha)$.

The Deconflictor problem formulation also needs constraints. In this work, constraints describe the physical operation of the test system. For example, equality constraints are used to ensure that the studied system obeys the relevant power flow equations. The implemented Deconflictor receives these constraints from one of the apps. As the Resilience App and the Decarbonization App both use the same system model, the Deconflictor can obtain its constraints from either app.

The mathematical formulation described above has been implemented using PuLP [21] in Python.

4.3.2 Resolution Vector Calculation

Once we formulate a Deconflictor-level MILP problem, we can proceed to solving it. We selected CBC [22] as our solver. The optimality gap was set to 0.01 and the time limit to 8s. The deconflicted commands for the batteries and the regulators, which are also known as Resolution Vector, can be directly obtained from the corresponding decision variables of the solution. Figure 7 shows the results for the power and SoC of Battery 2 for the weight, α , equal to 0.0, 0.5 and 1.0.

Even though the utility functions of the two apps are combined with equal weights, the results for Battery 2 seem to be very close to the $\alpha = 1.0$ case. This case is also known as Decarbonization

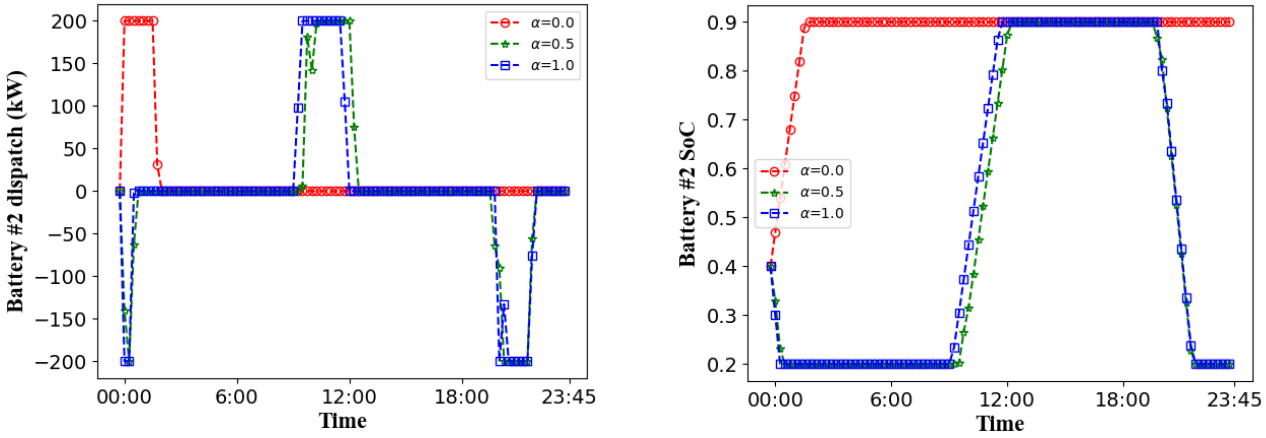


Figure 31: Optimization-based deconfliction results for battery power (left) and SoC (right) with the weight, α, equal to 0.0, 0.5 and 1.0.

Figure 32: Optimization-based deconfliction results for weight, α, equal to 0.5 (left) and 0.25 (right).Figure 33: Optimization-based deconfliction results for battery power (left) and SoC (right) with the weight, α, equal to 0.0, 0.5 and 1.0.

App exclusivity because this weight combination eliminates the Resilience App utility function. While Battery 2 is presented here as a representative example, the same conclusion can be reached by observing all other batteries of the system. A plot of the app utility functions alongside the Deconflictor-level objective function can help clarify this. Figure 8 (left) shows the objective function value, the Decarbonization App utility function value, and the Resilience App utility function value, respectively for $\alpha = 0.5$.

As the SoC of each battery is limited between 0.2 and 0.9 to ensure longevity, the solution of the combined problem shows that the optimization-based Deconflictor gets relatively little gain from its Resilience App component, which explains why the Decarbonization App utility function seems to dictate the Deconflictor's actions. In order to further investigate this hypothesis, our sensitivity analysis focused next on α values of 0.25 and 0.75. Figure 8 (right) shows the Deconflictor objective function value, the Decarbonization App utility function value, and the Resilience App utility function value, respectively for $\alpha = 0.25$.

In Figure 8 (right), the Resilience App utility function contribution seems to dominate the behavior of the Deconflictor, as expected due to the difference in weight. In order to verify this, Figure 9 shows the results for the SoC of Battery 2 for α equal to 0.0, 0.25 and 1.0, respectively. It can be clearly seen that the Resilience App indeed dominates the Decarbonization App for α equal to 0.25.

The main conclusion of this sensitivity analysis is that the normalization of utility functions is not enough. Seemingly unexpected results can be caused by the different ways that decision variables are related to each other through both the objective function and the constraints of the optimization problem. Therefore, users of the Deconflictor should make an extra effort to properly study its behavior based on the specific types of apps that it might encounter, and properly calibrate it before operations.

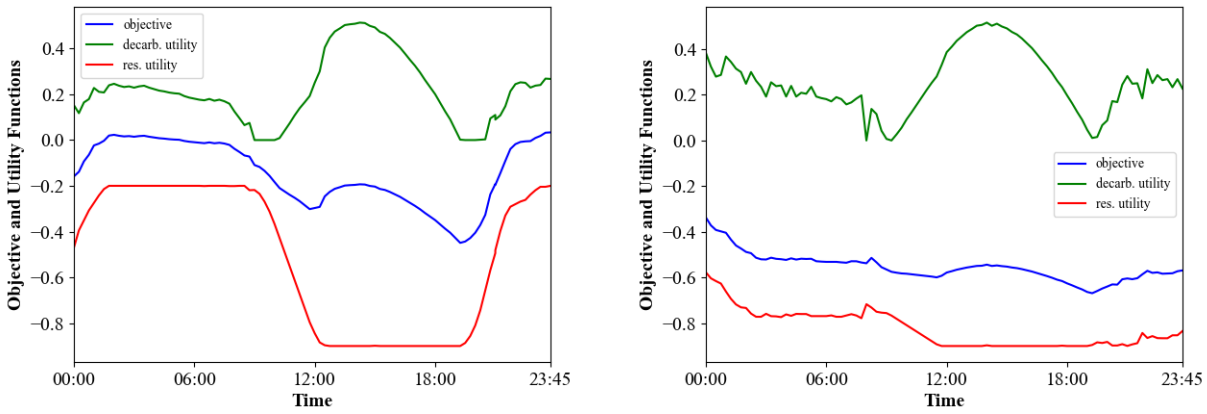


Figure 34: Optimization-based deconfliction results for weight, α, equal to 0.5 (left) and 0.25 (right).

Figure 35: Optimization-based deconfliction results for battery SoC with the weight, α, equal to 0.0, 0.25 and 1.0.Figure 36: Optimization-based deconfliction results for weight, α, equal to 0.5 (left) and 0.25 (right).

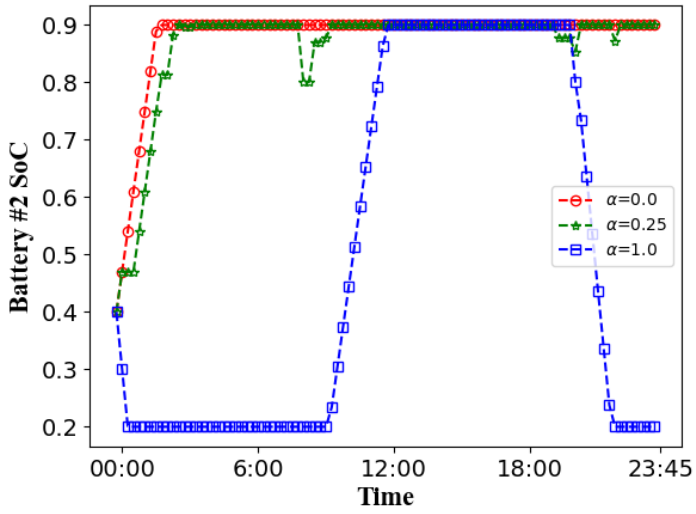


Figure 37: Optimization-based deconfliction results for battery SoC with the weight, α, equal to 0.0, 0.25 and 1.0.

Figure 38: Conceptual workflow for a unified deconfliction service within the GridAPPS-D Platform.Figure 39: Optimization-based deconfliction results for battery SoC with the weight, α, equal to 0.0, 0.25 and 1.0.

5.0 Alternatives Analysis

This section identifies desired functional characteristics of a GridAPPS-D deconfliction service and assesses the suitability of elements of each solution domain to implement the desired characteristics. This assessment is accomplished by mapping the desired deconfliction service characteristics to solution-element attributes. Attributes may be binary, qualitative, or quantitative. An alternatives analysis is performed for several elements from each solution domain evaluating whether and how-much each element possesses each attribute. A set of elements that are particularly well-suited for a deconfliction service is identified along with a subset of those elements that are compatible with each other.

The envisioned GridAPPS-D Deconfliction Service has six functional characteristics:

1) Solutions balance app objectives, subject to operator decisions and system constraints.

2) The environment of apps is dynamic: apps can join or leave the operational platform and do not need to conform to archetypes; messages can be asynchronous.

3) Apps are be supported regardless of whether and to what extent they are designed to actively participate in any deconfliction scheme.

4) Apps are black boxes from the platform perspective. The deconfliction service may define an interface but cannot assume that it will know internal logic or objectives of apps.

5) Solutions scale in a way that is compatible with the level detail in GridAPPS-D and a number of apps that will be present in advanced distribution operations.

6) A solution is guaranteed.

In addition to the overall functional characteristics listed above, it is possible to define a set of key attributes to quantify those requirements. The attributes are used to evaluate components of each solution technique and highlight strengths and weaknesses of components suitable for adoption into a combined solution.

Attributes of elements that support the deconfliction service characteristics:

- **Arriving / Departing Apps** (binary): This attribute is related to the dynamic app environment and evaluates whether a solution component is compatible with apps joining and leaving the deconfliction framework, possibly without any notification. Within the combined solution, the starting point is the Conflict Matrix such that when apps arrive and depart, new rows are added (or removed) from the Conflict Matrix).

- **Asynchronous Setpoints** (binary): This attribute is also derived from the dynamic app environment and evaluates whether a solution component is compatible with asynchronous messages. Apps will likely send setpoint messages at different times and different intervals,

and thus, it is important that the deconfliction service is able to handle asynchronous setpoints.

- **Requires Participation** (binary): This attribute describes whether participation by apps is required by the solution component to achieve desired results. It is anticipated that the deconfliction service cannot constrain app design to require participation, thereby excluding naïve applications.

- **Incentivizes Participation** (binary): Simultaneously, it is desirable for the deconfliction service to incentivize and possibly reward active participation in the deconfliction solution.

- **Balances App Objectives** (binary): To avoid the deconfliction service from simply overriding all applications or ignoring all objectives of an app in favor of those from another, it is important that the deconfliction solution balance objectives, subject to operator decisions and system constraints.

- **Requires App Objectives** (binary): Conversely, solutions that require all app objectives to be shared will be impact the ability of the deconfliction service to handle naïve apps and black box apps that do not share their objectives or internal formulation.

- **Scalability with Number of Apps** (scaling factor): As the number of apps increases, it is desirable that the overall deconfliction complexity remain constant or increase linearly with the number of apps sending setpoints for a given iteration.

- **Scalability with Number of Setpoints** (scaling factor): Similarly, as the number of controllable devices increases, it is desired that the solution complexity scale linearly.

- **Scalability with Distributed Decomposition** (scaling factor): This attribute captures two aspects of distributed decomposition. The first is scalability with the number of times the overall problem is decomposed into distributed control areas. The second is the increases in complexity as the number of distributed areas increases.

- **Guaranteed Solution** (binary): The deconfliction service is required to guarantee a final solution that can be translated in protocol-specific control messages. This attribute evaluates whether a solution component can (by itself) guarantee a solution result.

In Table 6. The components of each of the solution techniques described in Section 2 and illustrated numerically in Section 4 are evaluated for the attributes above. Scalability issues in Table 6 are indicated using big-O notation [23], which describes the function that bounds the computation time or memory needed for a given problem of size n. The concept of NP-hardness [24] is also used, which identifies whether the problem is at least as hard as the hardest problem in NP, which in turn, refers to whether the problem can be solved and verified in polynomial time (rather than exponentially increasing time).

Table 6: Alternatives Analysis of Deconfliction Solution Elements

	Arriving / Departing Apps	Asynchronous Setpoints	Requires Participation	Incentivizes Participation	Balances App Objectives	Requires App Objectives	Scalability with # Apps	Scalability with # Setpoints	Scalability with Distributed Decomposition	Guaranteed Solution
Device control budgets	***Yes***	***Yes***	***No***	***No***	***No***	***No***	***O(1)***	***O(n)***	***O(1)***	***No***
System-status based rules	***Yes***	***Yes***	***No***	***No***	***No***	***No***	***O(1)***	***O(n)***	***O(n)***	***No***
Decomposition by distributed area	Yes	Yes	No	Yes	Yes*	No	O(1)	$O(n^2)$	O(n)	Yes
Device-level decomposition	Yes	Yes	No	No	No	No	O(1)	O(n)	O(n)	Yes
Predictive snapshot power flows	Yes	Yes	No	No	No	No	O(1)	$O(n^2)$	O(1)	No
Use of generic deconfliction decision criteria	Yes	Yes	No	No	No	No	O(1)	O(n)	O(1)	Yes
Setpoint weighting/selection	Yes	Yes	Yes	Yes	Yes*	No	O(n)	O(n)	O(1)	No
Criteria weighting/selection	Yes	Yes	No	Yes	Yes*	No	O(1)	O(1)	O(1)	No
Ranking criteria based on real-time conditions	Yes	Yes	No	Yes	No	No	O(1)	O(1)	O(1)	Yes
App-to-app iterations before solving (e.g., blocking)	No	No	No	No	Yes	No	$O(n^2)$	O(n)	O(n)	Yes
Solves every time apps update “opening-bid” setpoints (e.g., continuous)	Yes	Yes	No	No	Yes	No	$O(n^2)$	O(n)	O(n)	Yes*
Mediator design considers app-characteristics (at app-design phase)	No	Yes	No	No	Yes	No	O(n)	O(n)	O(1)	Yes
Mediator supports unknown app characteristics	***Yes***	***Yes***	***No***	***Yes***	***Yes***	***No***	***O(n)***	***O(n)***	***O(1)***	***Yes***
Apps respond to contextual status signals	***Yes***	***Yes***	***Yes***	***Yes***	***Yes***	***No***	***$O(n^2)$***	***O(n)***	***O(1)***	***No***
Apps respond to incentive signals	Yes	Yes	Yes	Yes	Yes	No	O(1)	O(1)	O(n)	No
Using weight factors to incentivize flexibility	Yes	Yes	No	Yes	Yes	No	$O(n^2)$	O(n)	O(n)	No
Setpoint-informed optimization	***Yes***	***Yes***	***No***	***No***	***No****	***No***	***O(n)***	***O(n)****	***O(1)***	***Yes***
Penalty for distance from app targets (e.g., setpoints or utility functions)	No	Yes	Yes	Yes	Yes	Yes	O(n)	NP-H*	O(1)	No
Utility function	No	Yes	Yes	Yes	Yes	Yes*	O(n)	NP-H*	O(1)	Yes
Weighted utility-function	No	Yes	Yes	Yes	Yes	Yes*	O(n)	NP-H*	O(1)	Yes
App-provided constraints	No	Yes	Yes*	Yes	No	No	O(n)	NP-H*	O(1)	No
Use distributed optimization for distributed deconfliction	No	Yes	Yes*	Yes	No	No	$O(n^2)$	NP-H*	O(n)	No

5.1 Element Description and Discussion

Each key element of the three solution techniques is described in this section, with further details and reasoning for the attributes scores provided in Table 6. Elements in orange text will be included in the combined solution in Section 5.2.

5.1.1 Rules-Based Elements

The rules-based solution technique described in Section 2.1 and demonstrated in Section 4.1 contains nine core elements, which are scored in the first portion of Table 6.

- ***Device control budgets:*** Setpoints are constrained according to device lifecycle and asset health considerations. This component is compatible with most other elements, does not require sharing/cooperation of black-box apps, and scales linearly with all attributes.

- ***System operation rules:*** System status information is used to constrain setpoints that would violate system limits or operational practices. This component is compatible with most other elements but may need to be relaxed during abnormal operations to ensure a solution is found.

- *Decomposition by distributed area:* The deconfliction problem is decomposed using the Laminar Coordination Framework into local sub-problems, with each deconfliction agent only solving for setpoints in its area. An area-independence approximation is applied to limit consideration of conflicts per area; system-level criteria are considered on a per-area basis. Decomposition is critical for the predictive snapshots to reduce multiplicative growth of the solution space.

- *Device-level decomposition:* The deconfliction problem is solved in a fully decentralized manner with predictive snapshots and decision criteria calculated on a per-device basis. This approach is massively scalable but does not guarantee optimality.

- *Predictive snapshot power flows:* Snapshot power flows are used to predict outcomes of setpoints and assess decision criteria and violations of system rules (using distributed area or device-level decomposition as applicable).

- *Use of generic deconfliction decision criteria:* Candidate outcomes are evaluated using criteria selected by the DSO, with or without objectives shared by apps; criteria may be system-driven or app-objective-driven.

- *Setpoint weighting / selection:* Device setpoints are weighed according to operator- or app-driven inputs. Assignment of exclusive control of a given device to a specific app can be accomplished with setpoint weights, as long as exclusive access is applied narrowly and during abnormal conditions

- *Criteria weighting / selection:* Decision criteria are weighted according to operator- or app-driven inputs with participants able to specify preference for specific decision criteria.

- *Ranking decision criteria based on real-time conditions:* Criteria weighting and selection are adjusted in real time based on normal/alert/emergency grid conditions.

5.1.2 Cooperative Elements

The cooperative solution technique described in Section 2.2 and demonstrated in Section 4.2 contains seven core elements, which were evaluated in the second portion of Table 6:

- *App-to-app iterations before solving* (e.g., blocking): Apps are permitted to consider context iteratively and update setpoints before a solution is published.

- *Solves every time apps update "opening-bid" setpoints* (e.g., continuous): The Deconflictor tracks context-independent setpoints (i.e., opening bids) separate from context-informed setpoints and produces a solution from previous iterations each time an app updates its opening bid.

- *Mediator design considers app-characteristics* (at app-design phase): App interfaces and/or behaviors drive mediator design.

- ***Mediator supports unknown app characteristics*:** A mediator is designed around a specified app-to-mediator interface, allowing the mediator to support uncharacterized apps.

- ***Apps respond to contextual status signals*:** Apps may update setpoints based on context, e.g., setpoints generated by other apps or system status.

- *Apps respond to incentive signals:* Apps may update setpoints based on incentive signals produced by a mediator.

- *Using weight factors to incentivize flexibility:* Deconflictor design includes reward mechanisms that may influence app behavior.

5.1.3 Optimization Elements

The optimization-based approach to deconfliction described in Section 2.3 and demonstrated in Section 4.3 contains six core elements, which are presented in the third portion of Table 6. Scalability of all elements in this list (except the first which admits a closed form solution) with respect to number of setpoints is NP-hard because they require the solution of an optimization problem that includes discrete decision variables. Practical solvers usually terminate after satisfying some conditions such as identifying a local optimum point in poly time and/or an upper limit on execution time with a solution that is not strictly guaranteed to be globally optimal.

- ***Setpoint-informed optimization:*** Deconfliction is performed using optimization informed only by system-level information (if available) and setpoint requests produced by apps.

- *Penalty for distance from app targets* (e.g., setpoints or utility functions): Apps can provide functions expressing their dissatisfaction with deviations from setpoint requests (e.g., through monotonically decreasing functions for setpoint values above or below the requested value).

- *Utility function:* Apps can provide utility functions alongside their setpoint requests; the deconflictor can add these constraints to an optimization formulation to aid deconfliction.

- *Weighting of utility functions:* Utility functions provided by apps can be multiplied by operator-selected weights before being incorporated into a Deconflictor-level optimization problem.

- *App-provided constraints:* Apps can provide their constraints (usually called "app-provided constraint functions) alongside their setpoint requests; the Deconflictor can add these constraints to an optimization formulation to aid deconfliction.

- *Use distributed optimization for distributed deconfliction:* The Deconflictor can decompose its internal optimization problem into separate sub-problems for separate, clearly delimited areas of the grid.

5.2 Combined Solution

The following elements combine to produce a solution that has all of the GridAPPS-D deconfliction service functional characteristics.

- *Device control budgets:* Budgets can be used in the combined solution to reduce the size of the solution space by constraining system setpoints to those will not result in accelerated degradation of physical assets through frequent control actions and oscillating setpoints.

- *System operations rules:* Similarly, rules can constrain the solution space by eliminating setpoints that result in violations of system limits or operational best practices. The rules could be implemented as direct heuristics or constraints on an optimization problem.

- *Contextual status signals:* In the combined solution, status signals would be shared with or among apps such that they could update their desired setpoints based on the evolving context.

- *Mediator with support of unknown app characteristics:* The combined solution would include a mediator that incentivizes apps to come to a cooperative solution. The mediator in the combined solution would be able to support apps with unknown internal characteristics.

- *Setpoint-informed optimization:* As a fallback mechanism if a cooperative solution cannot be agreed upon by applications, the combined solution would solve an appropriately-distributed optimization problem that reflects the shared objectives of applications to determine an acceptable deconfliction solution.

It is anticipated that a GridAPPS-D deconfliction service combining the above elements (as shown in Figure 9) would satisfy all six core functional requirements identified:

1) *Solutions balance app objectives:* The combined solution would reflect the objectives of the apps through the use of mediation and optimization incorporating their objectives.

2) *The environment of apps is dynamic:* Both the individual elements and overall workflow of building the Conflict Matrix supports the ability for apps to join or leave without changing the structure or code of the deconfliction service.

3) *Apps can choose whether or not to participate:* The functional elements selected enable deconfliction of a combination of naïve and intelligent apps that are allowed to choose whether or not they want to respond to mediation signals and whether they want to share their objective functions.

4) *Apps are black boxes from the platform perspective:* None of the functional elements prescribe a particular manner in which the apps must be built. All interactions (including intercepting setpoints, offering mediation, and passing context signals) can be performed through a standards-based interface with consistent semantics and syntax.

5) Scalable for realistic ADMS apps: All of the selected elements scale either linearly or quadratically with the number of apps, devices, and distributed decomposition iterations.

6) A solution is guaranteed: The combination of elements guarantees that a deconfliction solution can be obtained through a set of rules, cooperation, and optimization that may be adjusted as the grid evolves through normal / alert / emergency operating conditions.

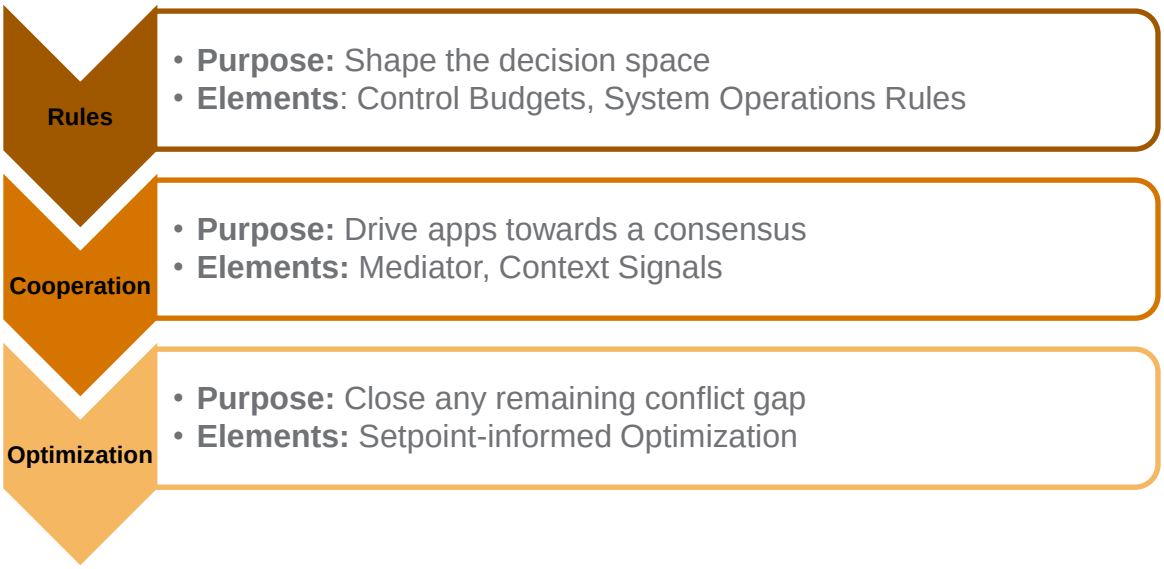


Figure 40: Conceptual workflow for a unified deconfliction service within the GridAPPS-D Platform.

6.0 Conclusion

This report provided an overview of the domain space and solution techniques that could be used to create a robust, flexible app deconfliction service. Three approaches were reviewed with summaries of the characteristics, elements, and results from preliminary demonstrations of solution techniques based on each approach. The first technique applied a combination of rules and heuristics to force a deconfliction solution that did not violate any rules for asset health and system operational constraints. The solution was chosen based on a set of predictive snapshot power flows used to calculate ranked utility function decision criteria formed from system-level objectives and optionally shared app objectives. The second technique applied a cooperative game theory strategy based on a combination of *press*, *compensate*, *advise*, and *ignore* contextual signals. This method applied these signals to drive applications to agree on a solution cooperatively through a mediator. The third technique combined the objective functions of each application into a global optimization problem. The problem was solved using utility functions supplied by each app with varying weights applied to each optimization goal shared by the apps.

The strengths and weaknesses of each solution technique were explored through a set of numerical demonstrations on modified IEEE 123 node and 9500 node test feeders. An alternatives analysis of individual deconfliction elements was performed with each solution technique element evaluated against criteria reflecting the dynamic app environment, need to balance app objectives, and scalability issues versus the number of applications, setpoints, and distributed control areas.

A combined solution integrating one to two elements from each solution technique was proposed. The combined solution would be implemented in the GridAPPS-D platform to achieve two prospective benefits 1) the open platform increases the total decision space available to apps by removing hidden restrictions on app functionality, operating conditions, and access to devices that effectively create orthogonal decision spaces for each app in order to avoid conflict; and 2) the open platform enables implementers to choose a set of best-of-breed apps rather than those paired with a vertically integrated solution. In this way, the open platform will tend to lead to device-setpoint conflicts as independently developed apps attempt to achieve their best outcomes by controlling as many devices as they can.

7.0 References

[1] A. P. Reiman, S. Poudel, G. Mukherjee, A. A. Anderson, O. Vasios, T. E. Slay, G. D. Black, A. Dubey and J. P. Ogle, "App deconfliction: Orchestrating distributed, multi-agent, multi-objective operations for power systems," *IEEE Access,* vol. 11, pp. 40314 - 40327, 2023.

[2] R. Melton, K. Schneider, E. Lightner, T. McDermott, P. Sharma, Z. Y. F. Ding, S. Vadari, R. Podmore, A. Dubey and R. Wies, "Leveraging standards to create an open platform for the development of advanced distribution applications," *IEEE Access,* vol. 6, pp. 37361 - 37370, 2018.

[3] A. Anderson, T. Wall, S. Vadari and A. Reiman, "Distributed rules-based deconfliction of ADMS applications: Part 1 Requirements & Decomposition (PNNL-34604-1)," Pacific Northwest National Laboratory, Richland, WA, 2023.

[4] A. Anderson, S. Vadari, T. Wall, P. Sharma and A. Reiman, "Distributed rules-based deconfliction of ADMS applications: Part 2 conceptual implementation (PNNL-34605-2)," Pacific Northwest National Laboratory, Richland, WA, 2023.

[5] A. Anderson, A. Fisher, S. Poudel, A. Reiman, T. Wall and S. Vadari, "SMARTER rules-based distributed deconfliction of ADMS applications," in *Submitted to IEEE Innovative Smart Grid Tech Conf.*, Washington DC, USA, 2023.

[6] W. Edwards and F. H. Barron, "SMARTS and SMARTER: Improved simple methods for multiattribute utility measurement," *Organizational Behavior and Human Decision Processes,* vol. 60, pp. 306 - 325, 1994.

[7] MESA Open Standards for Energy Storage, "MESA-ESS specification version 1.0," December 2018. [Online]. Available: http://mesastandards.org/wp-content/uploads/MESA-ESS-Specification-December-2018-Version-1.pdf. [Accessed March 2023].

[8] F. J. Rees and R. E. Larson, "Computer-aided dispatching and operations planning for an electric utility with multiple types of generation," *Power Apparatus and Systems, IEEE Trans. on,* vol. 90, no. 2, pp. 891 - 899, 1971.

[9] C. Peng, P. Xie, L. Pan and R. Yu, "Flexible robust optimization dispatch of hybrid wind/photovoltaic/hydro/thermal power system," *Smart Grid, IEEE Trans. on,* vol. 7, no. 2, pp. 751 - 762, 2016.

[10] W. Alshabi, S. Ramaswamy, M. Itmi and H. Abdulrab, "Coordination, cooperation and conflict resolution in multi-agent systems," in *Innovations and advanced techniques in computer and information sciences and engineering*, Netherlands, Springer, 2007, pp. 495 - 500.

[11] X. Castañer and &. N. Oliveira, "Collaboration, coordination, and cooperation among organizations: Establishing the distinctive meanings of these terms through a systematic literature review," *Journal of management,* vol. 46, no. 6, pp. 965-1001, 2020.

[12] O. Vasios, A. Riepnieks, T. Ramachandran and A. Reiman, "Optimization-based deconfliction of applications (PNNL-34175)," Pacific Northwest National Laboratory, Richland, WA, 2023.

[13] P. Shiva, G. Black, M. Mukherjee and A. Reiman, "Multi-objective power distribution operatons: Characterizing conflict and system volatility," *IEEE Access,* p. in press, 2023.

[14] A. A. Anderson, S. Vadari, J. Barr, S. Poudel, A. Dubey, T. McDermott and R. Podmore, "Introducing the 9500 Node Test System to support advanced power applications," Pacific Northwest National Laboratory, PNNL-33471, Richland, WA, 2022.

[15] A. A. Anderson, R. Podmore, P. Sharma, A. P. Reiman, R. A. Jinsiwale, C. H. Allwardt and G. D. Black, "Distributed application architecture and LinkNet topology processor for distribution networks using the Common Information Model," *IEEE Access,* vol. 10, pp. 120765 - 120780, 2022.

[16] Siemens Energy, "JFR single-phase voltage regulator," 2021. [Online]. Available: https://assets.siemens-energy.com/siemens/assets/api/uuid:8d0dea66-e1a6-4fd2-83ee-b69a4240a605/jfrsingle-phasevoltageregulator.pdf. [Accessed Sept 2023].

[17] Western Area Power Administration, "Proposed rates for firm power transmission ancillary services: Rate order no. WAPA-190," 2020. [Online]. Available: https://www.wapa.gov/regions/CRSP/rates/Documents/WAPA-190_Rates_Brochure_January_2020.pdf. [Accessed Sept 2023].

[18] US Energy Information Administration, "Colorado Electricity Profile 2021," Nov 2022. [Online]. Available: https://www.eia.gov/electricity/state/colorado/. [Accessed Sept 2023].

[19] Xcel Energy, "2022 solar bank election form," 2022. [Online]. Available: https://www.xcelenergy.com/staticfiles/xe-responsive/Working%20With%20Us/Renewable%20Developers/Solar-Bank-Election-Form.pdf. [Accessed Sept 2023].

[20] US Energy Information Association, "Carbon dioxide produced per kilowatthour of US electicity generation," Nov 2022. [Online]. Available: https://www.eia.gov/tools/faqs/faq.php?id=74&t=11. [Accessed Sept 2023].

[21] S. Mitchell, M. O'Sullivan and I. Dunning, "PuLP: A linear programming toolkit for Python," University of Auckland, Auckland, New Zealand, 2011.

[22] J. e. a. Forrest, "COIN-OR/CBC solver," [Online]. Available: https://doi.org/10.5281/zenodo.7843975.

[23] D. E. Knuth, "Big omicron and big omega and big theta," *SIGACT News,* vol. 8, no. 2, pp. 18 - 24, 1976.

[24] D. E. Knuth, "Postscript about NP-hard problems," *SIGACT News,* vol. 6, no. 2, pp. 15 - 16, 1974.

[25] T. E. Slay, M. Mukherjee, S. Poudel, G. D. Black and A. P. Reiman, "A framework for cooperation among power distribution system applications," *submitted to Energy,* vol. http://dx.doi.org/10.2139/ssrn.4523739, 2023.

[26] W. S. R. M. I. a. H. A. Alshabi, "Coordination, cooperation and conflict resolution in multi-agent systems," in *Innovations and advanced techniques in computer and information sciences and engineering, Springer* , Netherlands, 2007.

[27] A. P. Reiman, S. Poudel, G. Mukherjee, A. A. Anderson, O. Vasios, T. E. Slay, G. D. Black, A. Dubey and J. P. Ogle, "App deconfliction: Orchestrating distributed, multi-agent, multi-objective operations for power systems," *submitted to IEEE Access,* 2023.

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354

1-888-375-PNNL (7665)

www.pnnl.gov