

Lightweight, User-Directed Stochastic Resource Allocation

Submission Number: 239

Abstract

Stochastic resource allocation is a problem faced by most smart energy systems. Control algorithms must reason about imperfect models of competing user goals and assign a scarce and uncertain resource. Current approaches suffer from an inability to account for environment stochasticity and often make unrealistic assumptions about end user preferences. Methods that account for stochasticity are typically too computationally expensive to deploy on ultra low-cost hardware. We propose an approach to resource allocation that explicitly reasons about uncertainty in resource availability and end user preferences. Because of this ability, the algorithm is designed to interact directly with end users and provide quality of service guarantees in the face of uncertainty. Due to a compact modeling of the system's stochastic dynamics and a memory-efficient stochastic local search method, our approach is able to provide near optimal resource allocation policies and still run on low-cost hardware. While the algorithm is general, we investigate its application to resource allocation in developing world microgrids. The approach outperforms a simple thresholding method and compares well against a much more computationally intensive stochastic mixed integer linear program.

Introduction

Efficiently allocating scarce resources is a fundamental problem in modern energy systems. Utilities must cope with the increasing penetration of renewables and decide how to schedule conventional power plants. Within the home, smart appliances must determine when to consume power to match their owner's preferences. Outside the home, scheduling fleets of autonomous vehicles promises a reduction in emissions, but also challenges systems to meet the stochastic demand of passengers.

Much progress has been made in stochastic resource allocation for energy systems. Many approaches focus on stochastic mixed integer program formulations, which provide high quality solutions (Su, Wang, and Roh 2014b; Kim and Zavala 2016). However, because they typically require optimizing a large number of decision variables to account for uncertainty, they can be difficult to integrate with stochastic end users. Other approaches use market mechanisms where rational agents optimize utility functions to

ensure supply and demand are balanced. While these approaches work well in theory, human end users do not always act as utility-maximizing agents with well-defined utility functions (Kirschen 2003).

In this work, we present a method for integrating stochastic resource scheduling with end user preferences. We present the Chance-Constrained Stochastic Resource Allocation formalism. Through a compact representation of the stochastic state space, the algorithm can reason about uncertainty while avoiding large memory requirements. We solve the resource allocation problem through an innovative stochastic local search method, which also reduces memory overhead. The algorithm's output is a set of control policies that provide direct feedback to the end user.

Although resource allocation is necessary for many power systems, we focus on the example scenario of electrification in the developing world, where over 1.1 billion people lack reliable electricity (IEA 2013). This algorithm is part of a larger project to develop solar-powered microgrids for the developing world. A microgrid is a self-contained electricity grid that can connect a variety of loads and generation sources. Efficient resource allocation is critical in this use case. For example, resource allocation can ensure energy is available for critical activities such as lighting. It can greatly reduce the required battery size and system cost by effectively scheduling users. Because willingness to pay is only \$2-3 per month, cost is extremely important (Campanella 2013). The computational hardware is limited to \$2 microcontrollers, which operate with less than 1 MB of RAM. Furthermore, complex user interfaces are not possible and user intentions must be largely inferred.

Related Work

A traditional approach to resource allocation is using mixed integer linear programming (MILP) to schedule loads while minimizing a cost function (Parisio and Glielmo 2011; Tenfen and Finardi 2015; Morais et al. 2010). MILP approaches offer high quality solutions valid over fine time discretizations. However, MILP approaches are computationally-intensive and infeasible for the low cost microcontrollers used in our application. Furthermore, most solutions are designed for developed world microgrids, whose purpose is typically backup power and/or solving the scheduling problem at the utility level without end user interaction (Liang

and Zhuang 2014).

Distributed market-based approaches are also common in solving the power resource allocation problem. An innovative approach in (van den Briel, Scott, and Thiebaux 2013) uses local voltage signals as an indication of when appliances can turn on. Much of the work using market and pricing mechanisms builds on game theory. One example in (Samadi et al. 2012) uses an auction mechanism to optimize both user utility and grid utility. This and other market-based work relies on expert knowledge, such as users' exact objective functions or which loads are *controllable* and which *non-controllable*. In real life, this information is difficult and tedious to obtain. The literature that most closely aligns to this work is (Ono, Graybill, and Williams 2012), where user goals and uncertainty are taken into account to derive a control policy for the optimization of a home heating system. This algorithm provides reliability guarantees in the face of uncertainty.

Innovations

This work sits between techniques in stochastic programming and goal-directed, chance-constrained planning. We develop a stochastic resource allocation algorithm that receives statistical models of resource availability and demand and outputs a user-specific control policy. Specifically, we offer the following contributions:

1. The ability to reason directly over distributions that describe user goals allows direct integration with end users and the ability to directly shape user behavior, unlike most approaches in literature.
2. Modeling the stochastic evolution of the resource as a discrete time and state Markov chain allows a memory-efficient representation of state stochasticity.
3. A conflict-based stochastic local search method designed for resource allocation efficiently explores the search space and obtains solutions that are close to optimal.

Stochastic Resource Allocation Problem

The problem is formalized as a Chance-Constrained Stochastic Resource Allocation Problem (*ccSRAP*). A *ccSRAP* is an 8-tuple $P = \langle T, G, \mathcal{U}, \mathcal{A}, CC, SC, \Pi, f_u \rangle$ where:

- T is the planning horizon, with time discretized.
- G is a distribution describing resource availability or generation, defined over T : $\text{supp}(G) = \{t | 0 \leq t \leq T\}$.
- \mathcal{U} is a set of users.
- \mathcal{A} is a set of resource-consuming activities. Each activity a_i is associated with a user. Corresponding to each activity is a probability distribution $f_A(a)$ that describes the user's resource consuming preferences such as time-of-use or the amount of resource required.
- CC is a set of chance constraints that enforce the probability of serving certain critical activities. Each chance constraint CC_i is associated with one activity distribution $f_A(a_{i,cc})$. The set of activities associated with chance

constraints, \mathcal{A}_{cc} , is a subset of all activities $\mathcal{A}_{cc} \subseteq \mathcal{A}$, i.e. not all activities are associated with chance constraints.

- SC is a set of stochastic constraints that direct the evolution of the system state s , i.e. the availability of a resource over time.
- Π is a set of control policy functions, π_i , where each policy π_i is associated with an activity distribution $f_A(a_i) \in \mathcal{A}$. Each function π_i maps the system state s to a set of time bounds M_π that describe a_i 's allowed start time and duration: $M_{\pi,i} = \pi_i(s)$.
- f_u is an objective function that maps the set of policies to positive utility $U \in \mathbb{R}^+$, $U = f_u(\Pi)$.

Each component is described more fully in the following sections.

Model of Time

The algorithm reasons over a specified time horizon T where time is discretized into Δt . A natural time horizon to use for electricity scheduling is 24 hours because of the repetitive nature of generation and demand.

Input Model of Demand and Generation

The input is the joint distribution of resource demand D and resource availability or generation G :

$$f(G, D) \quad (1)$$

To simplify modeling of (1), independence is assumed between demand and generation. Further, the naive assumption is made of independence between individual users and their demanded activities:

$$f(G, D) = f(G) f(D) = f(G) \prod_i f_{A_i}(a_i) \quad (2)$$

To model $f(G)$, a sample average approximation (SAA) is used. Sample average approximation is commonly used in stochastic optimization problems and requires sampling resource scenarios from their distribution (Kim, Pasupathy, and Henderson 2011) where a single scenario i is a vector: $G_i = [g_{i,0}, g_{i,1}, \dots, g_{i,T}]$. The distribution can be generated through historical or sensor data, such as observing the solar generation available on a microgrid.

Activity Model of Demand

Resource demand is modeled in terms of activities that users would like to accomplish. Each activity and its distribution $f_A(a_i)$ is associated with a user. A is a vector of random variables that represents the user's desired start time, duration, and resource demand for the activity:

$$f_A(a_i) = f_A(\text{start_time}, \text{duration}, \text{resource}) \quad (3)$$

For the microgrid application, constant power direct current (DC) activities are assumed such as cell charging and lighting. Constant power is reasonable in the developing world context (Inam et al. 2015). In the case where users' preferences are uniformly distributed, (3) can be rewritten as a

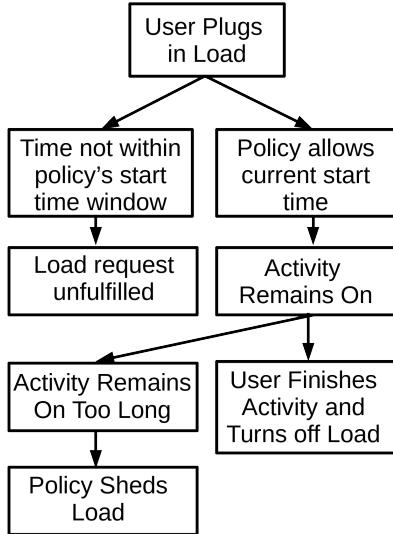


Figure 1: Depiction of the controller states in serving electrical activities for the microgrid use case

lower bound and upper bound on each of the uncertain quantities:

$$f_A(st, dur, r) \sim \mathcal{U}(t_{st,des}, t_{dur,des}, r) \quad (4)$$

$$\text{where } t_{st,des} \in [t_{st,lb,des}, t_{st,ub,des}] \quad (5)$$

$$t_{dur,des} \in [t_{dur,lb,des}, t_{dur,ub,des}] \quad (6)$$

$$r = r_{power} \text{ (a constant)} \quad (7)$$

A distribution of the form (4) is input to the algorithm as a model of the users' preferences. Because it represents the user's desires with regards to the start times and duration of the activity, it is referred to as the *preference distribution*. While this work assumes distributions with finite support (such as the uniform), distributions with infinite support can be truncated to match this model.

Output Control Policy

The algorithm outputs a set of control policies Π , which is a relationship between a state-action pair defined over the time horizon T . The overall policy Π is a set of independent policies for each individual activity i :

$$u_i(s) = \pi_i(s(t)) \quad \forall \pi_i \in \Pi \quad (8)$$

In the microgrid example, the state s is a function of the battery's state of charge b and the estimated amount of solar energy remaining, $\mathbb{E}[G]$. Each control policy is unique to an activity, while the state s is network-wide. The action $u_i(s)$ is whether to supply an activity with the requested resource C_{req} , depending on whether the resource is requested within precomputed time windows M_π :

$$u_i(s) = \begin{cases} C_{req} & \text{for } t \in M_{\pi,i} = [t_{lb,\pi}(s), t_{ub,\pi}(s)] \\ 0 & \text{for } t \notin M_{\pi,i} = [t_{lb,\pi}(s), t_{ub,\pi}(s)] \end{cases} \quad (9)$$

Eq. 9 is further divided into two separate policies $u_{start,i}$ and $u_{dur,i}$, controlling an activity's allowed start times and

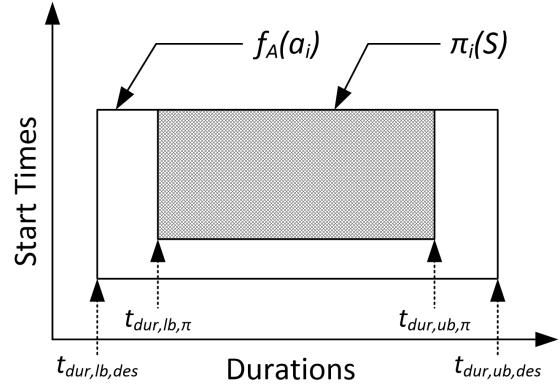


Figure 2: The objective seeks to minimize the difference between the policy π_i 's output time bounds and the user's preference distribution $f_A(a_i)$, shown here for a uniform distribution over start times and durations.

durations respectively:

$$u_{start,i} = \pi_{start,i}(b, \mathbb{E}[G], t) \mid a_i \text{ not started} \quad (10)$$

$$u_{dur,i} = \pi_{dur,i}(b, \mathbb{E}[G], t) \mid a_i \text{ started} \quad (11)$$

Eq. 10 represents the policy given that the activity has not yet started, i.e. it governs the activity's allowed start times conditioned on the network state. Eq. 11 governs the activity's duration. Creating policies for an activity's start time and duration provides more control authority. There may be cases when it is advantageous to shift an activity (i.e. change its start time) or other occasions when it is necessary to curtail its resource usage (i.e. change its duration). The control strategy is depicted in Fig. 1.

Objective Function

While many electricity resource allocation problems are formulated using objectives that focus on minimizing grid cost, this work proposes a user-centric objective. The objective seeks to maximize the probability that the output policy set Π allows users to receive a resource when and in the quantity that they want:

$$\max_{\Pi} \sum_{\mathcal{A}} \sum_{supp(f_A(a_i))} P(a_i \in M_\pi | A_i = a_i, \pi_i) P(A_i = a_i) \quad (12)$$

$$- c_1 \sum_{crit.} \mathbb{1}_{U_a \leq U_{a,min}} \quad (13)$$

Maximizing the Likelihood of Achieving User Goals

The first term, (12), seeks to determine the policy that maximizes the likelihood of achieving user goals, conditioned on users' preferences and the control policy, π_i . For a grounded example, assuming a user wants 5W of power starting between 1 PM and 3 PM with a duration between 2 to 3 hours, the best policy is one that allows all start times and durations over the distribution's support with probability 1 (i.e.

the user may start whenever they want and consume their desired amount of electricity almost surely). The approach is illustrated in Fig. 2.

The second term, $P(A_i = a_i)$, indicates that utility is weighted by the preference distribution $f_A(a_i)$, i.e. the probability the user chooses a specific realization of A_i . The inner summation is over the support of the user's preference distribution (no utility is gained by providing a resource when a user does not want it). The first term, $P(a_i \in M_\pi | A_i = a_i, \pi_i)$, is the probability a specific activity realization is successful conditioned on the policy. It is more difficult to compute. First, the term is conditioned on the resource's distribution, $P(s)$:

$$\begin{aligned} P(a_i \in M_\pi | A_i = a_i, \pi_i) &= \\ \int_s P(a_i \in M_\pi | A_i = a_i, \pi_i, s) P(s) ds &\approx \\ \sum_S P(a_i \in M_\pi | A_i = a_i, \pi_i, S) P(S) \end{aligned} \quad (14)$$

Where the approximation to the integral is made possible by a discretization of the continuous state s . To compute Eq. 14, it is necessary to determine $P(S)$, which is covered in the following section.

Chance Constraints on Activity Reliability The second term in the utility, line 13, is a penalty term for not serving a minimum utility to activities contained within CC . CC is a set of pre-specified critical activities that should be served with a minimum reliability. This is useful to provide quality of service guarantees in the face of stochastic resource availability. A chance constraint on a critical activity takes the form:

$$P(U_{a,crit,min}) \geq 1 - \epsilon \quad \forall a \in CC \quad (15)$$

where $U_{a,crit,min}$ is a minimum utility that activity a is guaranteed with probability $1 - \epsilon$. Because probability mass serves as a proxy for utility, Eq. 15 is a guarantee on the probability mass a critical activity receives. It is written generally because there are many possible forms the minimum utility, $U_{a,crit,min}$, could take. For example, reliability guarantees may be written with respect to activity start times, durations, or both. For this work, we assume reliability constraints are specified with regards to an activity's duration (such as guaranteeing two hours of lighting with 90% reliability over all desired start times). This specification can be translated into the minimum utility:

$$\begin{aligned} U_{a,crit,min} &= P(a_{crit,min}) = \\ \sum_S P(dur \geq dur_{min} | S, \pi) P(S | \pi) &\geq 1 - \epsilon \end{aligned} \quad (16)$$

The minimum utility requirement, Eq. 16, is included in the objective function as a barrier term and ensures the solution incurs a penalty for each violated chance constraint. The reason that constraint violation is included as a barrier term and not a hard constraint is that the algorithm must always return a solution. It may be impossible to satisfy Eq. 15 (for example, if solar generation is constantly poor during a rainy season). In this case, the algorithm will maintain the reliability guarantees on a best effort basis.

Approximating Complex Scenarios as an MDP

To compute the utility of a policy, it is necessary to compute the distribution of the system state as it evolves in time. An efficient method to model stochastic state evolution is a discrete time and state Markov Decision Process (MDP). The MDP formalism consists of the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$: states \mathcal{S} , actions \mathcal{A} , transition probabilities \mathcal{T} , and a reward function \mathcal{R} . The MDP formalism is used to model the *ccSRAP*; the components are described in the following subsections.

States \mathcal{S} : Resource Availability Because of the vast number of generation and demand scenarios, modeling and reasoning about the system state is challenging. The Markov assumption combined with a state discretization makes this reasoning task much more tractable. Instead of reasoning about a large number of resource scenarios, it is necessary to consider the resource state at time $t - 1$:

$$P(S_{t+1} | S_t, S_{t-1}, \dots, S_{t-0}) = P(S_{t+1} | S_t) \quad (17)$$

To map from scenarios to discrete states, a discretization is introduced. Each continuous resource state falling within a predetermined bound, $\{s \in \mathbb{R}^n | s_{lb} \leq s < s_{ub}\}$, is mapped to a discrete state $S_{t,i}$. As mentioned above, in the microgrid case, the resource state is the amount of battery energy and expected solar generation available at time step t . Each discrete state S represents a set of continuous battery energies and solar powers, $S_{t,i} = \{b_t, \mathbb{E}[G_t] | b_{lb,i} \leq b_t < b_{ub,i}, G_{lb,i} \leq \mathbb{E}[G_t] < G_{ub,i}\}$.

Transitions Probabilities \mathcal{T} : Changing Resource States

To compute $P(S_{t+1,j} | S_{t,i})$, the probability of moving from one discretized resource state to another, it is necessary to consider all combinations of activities $a_i \in \mathcal{A}$ and generation scenarios that would cause a transition from $S_{t,i}$ to $S_{t+1,j}$. For the microgrid case, the transition probability is computed as

$$P(S_{t+1,j} | S_{t,i}) = P(G_{i,j}) P(b_{i,j} | \mathcal{A}, G_j) \quad (18)$$

In Eq. 18, $P(G_{i,j})$ is the probability of transitioning from one solar generation scenario to another, i.e. moving from a sunny day to a cloudy day. The second term, $P(b_{i,j} | \mathcal{A}, G_j)$, is the probability of the battery state b changing due to resource generation and consumption. It is computed as:

$$P(b_{i,j} | \mathcal{A}, G_j) = \sum_{\mathcal{E} \in C_{i,j}} P(\mathcal{E}) \quad (19)$$

where \mathcal{E} represents a single scenario of demand and generation that causes a transition from b_i to b_j and $C_{i,j}$ is the set of all such scenarios (there are potentially many such scenarios since the state is discretized). The scenario \mathcal{E} is described by the following constraints:

$$b_{t+1,j} = b_{t,i} + \sum_{\mathcal{A} \in \mathcal{E}_i} p(a_{t,i}) + g_{i,t} \quad (20)$$

$$\text{s.t. } b_{t,i,lb} \leq b_{t,i} < b_{t,i,ub} \quad (21)$$

$$b_{t+1,j,lb} \leq b_{t+1,j} < b_{t+1,j,ub} \quad (22)$$

where (20) is the energy balance, $p(a_{t,i})$ the power consumed by activity a_i at time t , and $g_{t,i}$ the energy generation.

Actions \mathcal{A} : Altering Activity Time Windows Unlike traditional MDPs, we cannot directly affect $P(S_{t+1,j}|S_{t,i})$ because we do not have direct control over resource generation or consumption. As indicated by Fig. 1, users are free to demand resources when they choose, but the policy enforces time windows that prevent consumption outside of those windows. The controller's actions affect the probability that a user is consuming power at time t , i.e. the policy affects the probability of scenario \mathcal{E} . This distribution is the probability of the generation scenario g_i multiplied by the probabilities of activities $a_i \in \mathcal{E}$ consuming the resource at time t :

$$P(\mathcal{E}) = P(g_i) \prod_{a_i \in \mathcal{E}_i} P(a_{t,i}|\pi) \quad (23)$$

The probability of a consuming resource at time t can be written in terms of the sums:

$$P(a_{t,i}|\pi) = \sum_{t_0, st} \sum_{t_0, dur} P(t_{dur, des} | t_{st, des}, \pi) P(t_{st, des}|\pi) \quad (24)$$

where the outer sum is over all possible policy-allowed start times that would result in the activity being on at time t and the inner sum is over all possible policy-allowed durations that would result in the activity consuming the resource at time t . Clearly, a policy that allows more start times and durations cannot decrease $P(a_{t,i}|\pi)$, a fact which is used later in designing the stochastic local search algorithm.

Rewards \mathcal{R} : States with Positive Resource For resource allocation problems, demand can be served when resources are available. For electricity scheduling, this means that transitioning to a state with positive battery energy receives a reward. The expected reward over all states is written in terms of this:

$$\mathbb{E}(\mathcal{R}) = \sum_t \sum_{S_t} \sum_{\mathcal{A}} U(a, \pi) P(S_t) \mathbb{1}_{b_t > 0} \quad (25)$$

where $\mathbb{1}_{b_t > 0}$ is an indicator function which is 1 if the state has positive battery energy and 0 otherwise. $U(a, \pi)$ is the utility gained by users being able to receive resource in state S_t . From Eqns. (12, 14), it is equal $P(a_i \in M_\pi | A_i = a_i, \pi_i, S) P(A_i = a_i)$, i.e. users only gain utility when the policy allows it and more utility is gained when the user is more likely to desire the resource.

Recurrent Planning Horizon

To fully specify the MDP, a planning horizon T must be specified. One approach is to impose an initial distribution on states and determine the optimal policy for all states less than T time steps into the future. We observe that in certain domains, such as electricity scheduling, demand is highly cyclical (Laloux and Rivier 2013). Most activities repeat on a daily or weekly frequency. We use this to compute the long term probability that the system is in state S_i over a repeating time horizon T_r . The state distribution at t_0 is the same as t_T , $S_{0,i} = S_{0,T_r}$. This creates a recurrent Markov chain with the transition matrix defined by \mathcal{T} . To obtain the long

Algorithm 1: Conflict-Based Stochastic Local Search.

```

Input: Model of generation,  $G$ 
Input: Set of distributions describing activities,  $D$ 
Output: Policy  $\Pi$  for all activities
1 for max number of iterations do
2    $pol\_proposed \leftarrow InitializeBasePolicy(D)$ 
3    $incumbent\_policy \leftarrow pol\_proposed$ 
4    $cur\_utility \leftarrow CalcUtility(pol\_proposed)$ 
5   foreach  $\pi_{i,j}$  in  $\Pi$  do
6     while  $\pi_{i,j}.IsImprovable()$  do
7       foreach  $a$  in  $A$  do
8          $pol\_proposed.Expand(i, j)$ 
9          $new\_utility \leftarrow$ 
10         $CalcUtility(pol\_proposed)$ 
11        if  $new\_utility < cur\_utility$  then
12           $pol\_proposed.record\_conflict()$ 
13           $pol\_proposed.rollback()$ 
14        else
15           $cur\_utility \leftarrow new\_utility$ 
16
17   if  $CalcUtility(pol\_proposed) \geq$ 
18      $CalcUtility(incumbent\_policy)$  then
19        $incumbent\_policy \leftarrow pol\_proposed$ 

```

term state probability, the Markov chain's stationary distribution is computed. This is done using (with a slight abuse of notation for π , in this case the stationary distribution π^s):

$$\pi^s = \pi^s \mathcal{T} \quad (26)$$

subject to:

$$\sum_{S_t} \pi_t^s = 1 \quad \forall t \in T \quad (27)$$

This system of linear equations can be solved efficiently using the conjugate gradient method or LU decomposition (Kreyszig 2010). Solving π_t^s for all time steps results in the long term probability that the system is in state $S(b, \mathbb{E}[G], t)$.

Conflict-Based Stochastic Local Search

A policy must be output that maximizes terms 12 and 13. Instead of using a computationally-intensive solver, a conflict-based stochastic local search strategy is used. The approach generates a policy keeping track only of the transition matrix, current policy, an incumbent best policy, and a set of conflicts. This allows generation of a good solution in a very memory-limited environment.

The insight into the search problem is that poor policies are easy to generate and can serve as bases for better policies. A further insight is that poor policies can be improved simply by increasing the policy's time bounds. This suggests the following simple strategy for generating a good policy: first, start with a set of policy bases Π_b , one for each activity. Second, incrementally improve the bounds of each policy until it is no longer possible to do so. The following theorem, stated without proof, guides this strategy:

Theorem 1. Given a set of activities $a_i \in \mathcal{A}$ and a set of base policies Π_b , increasing the time bounds of any individual activity monotonically decreases the distance between the utility of Π_b and the Pareto optimal frontier, with Pareto optimality defined in terms of the utilities of each a_i .

The theorem is guided by the fact that improving the time bounds of one activity's policy will never decrease the utility obtained by that activity. It may decrease the utility of other activities by reducing resource availability. Because of this, after each incremental improvement, the policy's global utility is calculated. If the utility has decreased, a conflict is extracted, which prevents further exploration of this region of the state space. To explore the state space, the search is made stochastic by randomizing which time bounds are expanded at each iteration. Finally, because the local search can become stuck in local optima, the inner search is embedded in an outer loop which maintains an incumbent best policy. The complete algorithm is provided in Alg. 1.

The inner loop first initializes base policies, which take the form: $\pi_{st,i}^0 = t$, s.t. $s_{lb} \leq t \leq s_{ub}$, $\pi_{dur,i}^0 = \delta t$. The initial start time policy is one start time randomly drawn from the preference distribution. The initial duration is the minimum possible duration, i.e. one discrete time step. Given initial policies $\pi_{st,i}^0$ and $\pi_{dur,i}^0$, local improvements are performed that either add or subtract a time step to the proposed policy's allowed start times or duration:

$$t_{st,lb,k+1} = t_{st,lb,k} - \Delta t \quad (28)$$

$$t_{st,ub,k+1} = t_{st,ub,k} + \Delta t \quad (29)$$

$$t_{dur,ub,k+1} = t_{dur,ub,k} + \Delta t \quad (30)$$

The goal is to improve each policy π_i such that the global utility of the policy Π increases at every iteration. If the global utility does not increase, a conflict is extracted. A conflict is a constraint that prevents exploring the search region that caused a decrease in utility. Conflicts guide the proposed policy until they prevent any further expansions or the proposed policy reaches the preference distribution's boundary. When the proposed policy can no longer be improved, its utility is compared against the incumbent and a new iteration begins.

During implementation, further search optimizations are used such as exploring the lowest energy states first and only updating the region of the transition matrix affected by an expansion. Due to space limitations, they will not be covered here.

Benchmarks

The Conflict-Based Stochastic Local Search is benchmarked against two other approaches: a simple thresholding approach and a stochastic MILP.

Rule-based Demand Response

One simple demand response approach is to use thresholds. To benchmark, we test thresholds that divide the activities into *critical* and *non-critical* activities. The simple demand response mechanism then results in the following rules:

- If battery charge $\leq x_1$, shed non-critical activities currently active

- If battery charge $\leq x_2$, shed critical activities currently active

where $x_1 > x_2$. While simple, the chief difficulty with rule-based approaches is determining the thresholds, x_1 and x_2 . For this work, x_1 and x_2 were set to 0.3 and 0.2 respectively, which provided good empirical performance among thresholds.

Scenario-Based MILP Approach

Because of their widespread use in literature, particularly power systems scheduling, a second benchmark is performed against a stochastic mixed-integer linear program (MILP). Many forms of stochastic MILPs are used and while it is difficult to benchmark against exactly the same formulation due to researchers' various requirements (such as transmission constraints, generation constraints, economic/market constraints), we construct a benchmark using some of the most salient features.

A challenge in stochastic MILPs is the representation of uncertainty. A popular method is to sample from the relevant distributions and reason over a finite number of scenarios (Nowak and Römisch 2000). This presents the challenge of quickly becoming intractable for multistage decision problems and further approximations are often deployed. One approximation is that of lumped demand. While our proposed method derives a policy for each individual activity, many approaches output a policy only for the aggregate resource demand signal (Varaiya, Wu, and Bialek 2011; Parvania and Fotuhi-Firuzabad 2010; Su, Wang, and Roh 2014a). This is a reasonable approximation for large grids but two issues may arise: because we are unable to reason about individual loads, we are unable to reason separately about critical loads. Many microgrids operate with various critical loads such as lights or cell towers (Almqvist and Rao 2015). To compare the MILP against our chance-constrained model, we use binary decision variables to model servicing critical loads and lump all other non-critical demand into an aggregate signal.

The program used to optimize the lumped demand stochastic MILP model is:

$$\min \sum_t \sum_s \ell_s[t] P(s) \quad (31)$$

$$\text{s.t. } b_s[t+1] = b_s[t] + g_s[t] - \sum_{\mathcal{A}_{crit}} d_{crit,a,s}[t] \quad (32)$$

$$- d_s[t] + \ell_s[t] \quad (33)$$

$$0 \leq b_s[t] \leq b_{max} \quad (34)$$

$$0 \leq g_s[t] \leq G_s[t] \quad (34)$$

$$\sum_s y_{crit,a,s} P(s) \geq 1 - \alpha, \quad y_{crit,a,s} \in \{0, 1\} \quad (35)$$

$$\bar{p}_a y_{crit,a,s} - d_{crit,a,t} = 0 \quad (36)$$

$$\forall a, t_{dur,min} \leq t \leq t_{dur,max}$$

The objective (line 31) seeks to minimize the aggregated expected loss of load. In (31), ℓ_t represents the total loss of load at time step t and $P(s)$ is the probability of scenario s . Constraint (32) ensures energy is balanced across time steps:

Type	St_{lb}	St_{ub}	D_{ub}	P (W)	Crit.
“Fan”	14	17	2	12	No
“Cell Charging”	6	12	3	5	No
“Lighting”	18	20	3	10	Yes

Table 1: Example activities used in the tests

Panel Size (W)	Battery (Ahr)	Chance Constr.
250	200	0.90

Table 2: Base generation, storage, and chance constraint parameters for the benchmark tests

$b_s[t]$ is the battery state of charge at time t in scenario s , $g_s[t]$ is the consumed generation, $d_s[t]$ is the lumped demand, and $d_{crit,a,s}[t]$ the power consumed by critical activity a . Constraint (33) ensures the battery remains within its capacity and (34) allows the model to waste unused generation: $G_s[t]$ is the actual generation at time step t .

Constraints (35, 36) relate to the chance constraint on critical activities. Variable $y_{crit,a,s}$ indicates whether critical activity a has started in scenario s . Line 35 ensures that the critical activity occurs in at least $1 - \alpha$ scenarios. Line 36 ensures that if a critical activity turns on, it consumes power (denoted by variable $d_{crit,a,t}$) for all time steps in its required duration. In line 36, \bar{p}_a is a constant describing activity a 's power demand. To mirror our MDP implementation, we change the formulation slightly and include satisfaction of the chance constraint as a penalty term in the objective:

$$\min \sum_t \sum_s \ell_s[t] P(s) + \gamma \sum_i Y_i \quad (37)$$

$$\sum_s y_{crit,a,s} P(s) - (1 - \alpha) + MY_a \geq 0, Y_a \in \{0, 1\} \quad (38)$$

where Y_a is a variable that indicates whether the chance constraint for activity a is satisfied. Constraint 38 replaces 35 and uses big-M notation to ensure that $Y_a = 1$ if the chance constraint is not met.

Regarding modeling the stochastic generation and demand scenarios s , there is a trade off between the MILP's accuracy and complexity. Similar to (Bouffard and Galiana 2008), three scenarios are used denoting times of low, medium, and high demand. To reduce the number of total scenarios, our scenario tree branches at 6 hour intervals for 4 levels over a 24 hour period. Even this approach leads to a scenario tree of size $3^4 = 81$.

Results

The Conflict-Based Stochastic Local Search (CB-SLS) approach is benchmarked against the threshold and MILP approaches. Test cases of various complexity and microgrid over/undersupply were performed. Example electrical activities for the test cases are shown in Table 1, which are typical loads for the developing world DC microgrid case. The desired start times and durations were randomized across tests. Lighting activities were marked as critical; each lighting activity was required to receive a 2 hour minimum dura-

tion 90% of the time. Table 2 shows other problem parameters such as the solar panel and battery size. The generation source's distribution was generated from the output of a 250 W solar panel, using sensor data collected near Jamshedpur, India.

The results of select tests are shown in Figures 3, 4, 5, and 6. For the MILP and CB-SLS algorithms, a policy was derived using the respective approach and then this policy was simulated over a large number of days. All reported utilities are from simulations of the output policies. Figure 3 was generated by running tests for various cases of over/undersupply for a 50 activity microgrid. Each plotted point represents a different test (i.e. for a test where the average solar to demand ratio is 0.8, the utility for the CB-SLS algorithm is approximately 0.81). A higher solar to demand ratio represents a more oversupplied grid and an easier problem. From 3, it is evident that the CB-SLS approach is close to that of the MILP over a wide range of grid conditions. Both approaches outperform the threshold approach. Figure 4 compares the algorithms' abilities to satisfy the chance constraints for a wide variety of grid conditions. Once again, the MILP and CB-SLS approaches outperform the thresholding approach with the CB-SLS showing slight improvement over the MILP approach. A reason for this could be the fact that our MILP models only single start times for critical activities; modeling multiple possible start times would require disjunctive constraints and make the model considerably more complex. Fig. 5 presents the algorithm scaling to a grid with 110 activities. Results are similar to those of the 50 activity test cases, with CB-SLS performing closely to the MILP.

The CB-SLS relies on discretizing the continuous state space and it is important to consider the accuracy of this discretization. Fig. 6 presents the mean squared error between our method's Markov chain distribution and the actual simulated distribution. As the number of states becomes large, the error is approaches zero. This presents the trade-off between accuracy and memory complexity: as more modeling accuracy is required, more memory is necessary to reason over more complex scenarios.

Memory Complexity

One of this work's main design constraints is that it should be implemented on low cost microcontrollers for developing world electrification. The state transition matrix is the most memory-intensive element. This has memory complexity $O(T * S^2)$ where T is the length of the time horizon and S is the number of discrete states at each time step. For a typical case, the time horizon has 24 steps and 9 states are used at each step for a complexity of approximately 2,000 floating point numbers. This yields approximately 8 kilobytes (4 bytes per float), which is feasible on a \$2 microcontroller with approximately 500 kb memory. This contrasts with the MILP approach. Fig. 7 plots the number of constraints required for example test cases modeled using 2 and 4 stage scenario trees. The 4 stage MILP requires tens of thousands of constraints for a modest number of activities. Solution of this MILP using a cutting-plane method and simplex tableau costs many megabytes of memory. While both

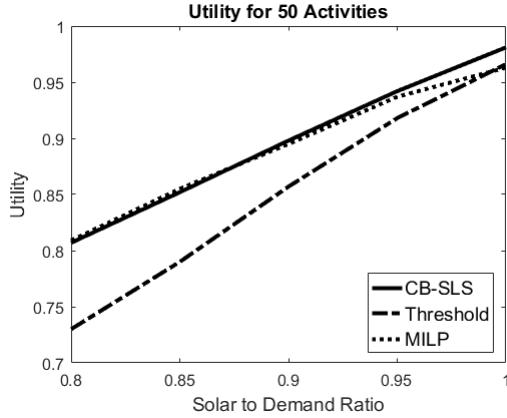


Figure 3: Utility as a function of the solar to demand ratio for all three approaches for the 50 activity test cases

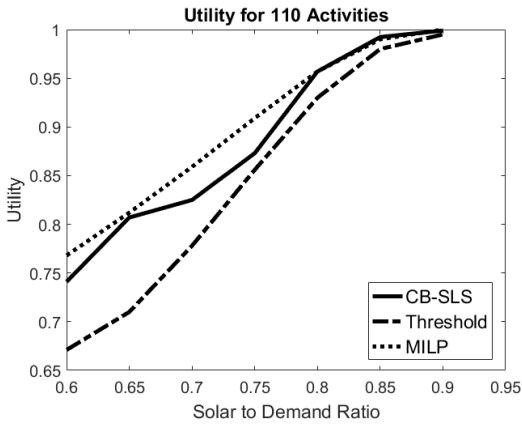


Figure 5: Utility as a function of the solar to demand ratio for the 110 activity test cases

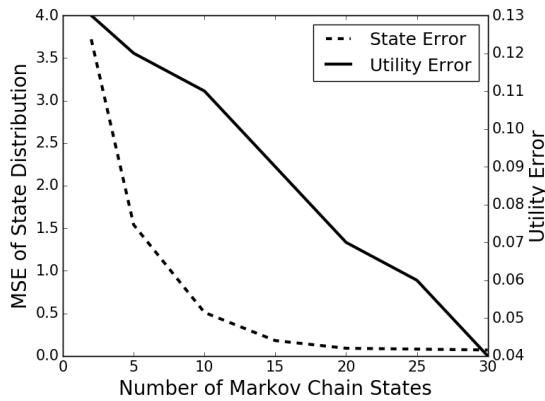


Figure 6: Mean squared distribution error and the utility error in approximating the full stochastic scenarios by a Markov chain as a function of the number of Markov chain states

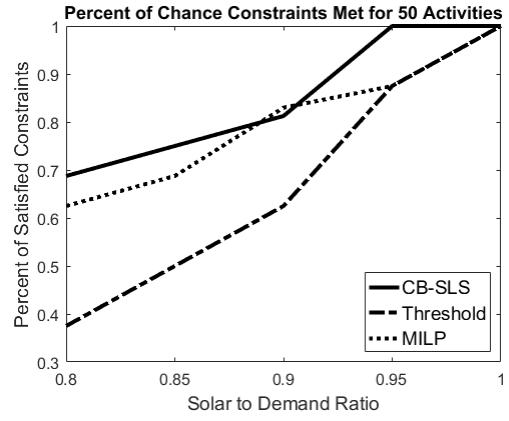


Figure 4: Percent of satisfied chance constraints as a function of the solar to demand ratio for the 50 activity test cases

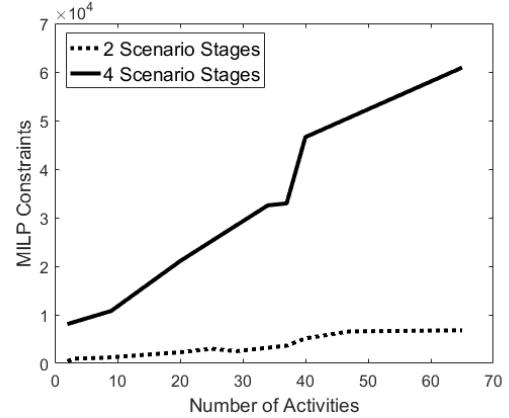


Figure 7: Constraints required for 2 and 4 stage MILPs as a function of the number of activities

the CB-SLS and MILP approach offer similar performance, the CB-SLS's compact representation of stochasticity allows it to be more easily implemented on embedded systems.

Conclusion

This work develops a lightweight, user-centric stochastic resource allocation algorithm. The algorithm is able to reason directly over distributions of demand and generation and avoid an exponential increase in complexity by modeling the stochastic state through a discrete time and space Markov chain. Despite the state discretization, the approach is able to approximate the continuous state well and compares well against a simple thresholding method and stochastic MILP. Further improvements can be made in how the discretization is constructed as certain discrete states contain little probability mass. Additional improvements lie not only on the algorithmic side, but also in its integration with end users. Future work is planned in our microgrid testbed where humans' responses will be the ultimate benchmark of the approach.

References

- Almqvist, P., and Rao, S. N. 2015. The quest to bring power, everywhere in india. [Online; posted 09-December-2015].
- Bouffard, F., and Galiana, F. D. 2008. Stochastic security for operations planning with significant wind power generation. *IEEE Transactions on Power Systems* 23(2):306.
- Campanella, A. 2013. An analysis of the viability and competitiveness of dc microgrids in northern india. Master's thesis, Massachusetts Institute of Technology.
- IEA. 2013. *World Energy Outlook*. Web: International Energy Agency.
- Inam, W.; Strawser, D.; Afidi, K.; Ram, R.; and Perreault, D. 2015. Architecture and system analysis of microgrids with peer-to-peer electricity sharing to create a marketplace which enables energy access. *9th International Conference on Power Electronics*.
- Kim, K., and Zavala, V. M. 2016. *Large-Scale Stochastic Mixed-Integer Programming Algorithms for Power Generation Scheduling*. Cham: Springer International Publishing. 493–512.
- Kim, S.; Pasupathy, R.; and Henderson, S. 2011. A guide to sample-average approximation. Technical report.
- Kirschen, D. S. 2003. Demand-side view of electricity markets. *Power Systems, IEEE Transactions on* 18(2):520–527.
- Kreyszig, E. 2010. *Advanced Engineering Mathematics*. John Wiley & Sons.
- Laloux, D., and Rivier, M. 2013. *Technology and Operation of Electric Power Systems*. London: Springer London. 1–46.
- Liang, H., and Zhuang, W. 2014. Stochastic modeling and optimization in a microgrid: A survey. *Energies* 7(4):2027.
- Moraes, H.; Kdr, P.; Faria, P.; Vale, Z. A.; and Khodr, H. 2010. Optimal scheduling of a renewable micro-grid in an isolated load area using mixed-integer linear programming. *Renewable Energy* 35(1):151 – 156.
- Nowak, M. P., and Römisch, W. 2000. Stochastic lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty. *Annals of Operations Research* 100(1):251–272.
- Ono, M.; Graybill, W.; and Williams, B. C. 2012. Risk-sensitive plan execution for connected sustainable home. In *BuildSys '12 Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, Toronto, Canada, November 6, 2012*, 45–52.
- Parisio, A., and Glielmo, L. 2011. A mixed integer linear formulation for microgrid economic scheduling. In *Smart-GridComm*, 505–510. IEEE.
- Parvania, M., and Fotuhi-Firuzabad, M. 2010. Demand response scheduling by stochastic scuc. *IEEE Transactions on Smart Grid* 1(1):89–98.
- Samadi, P.; Mohsenian-Rad, H.; Schober, R.; and Wong, V. 2012. Advanced demand side management for the future smart grid using mechanism design. *Smart Grid, IEEE Transactions on* 3(3):1170–1180.
- Su, W.; Wang, J.; and Roh, J. 2014a. Stochastic energy scheduling in microgrids with intermittent renewable energy resources. *IEEE Transactions on Smart Grid* 5(4):1876–1883.
- Su, W.; Wang, J.; and Roh, J. 2014b. Stochastic energy scheduling in microgrids with intermittent renewable energy resources. *IEEE Trans. Smart Grid* 5(4):1876–1883.
- Tenfen, D., and Finardi, E. C. 2015. A mixed integer linear programming model for the energy management problem of microgrids. *Electric Power Systems Research* 122:19 – 28.
- van den Briel, M.; Scott, P.; and Thiebaux, S. 2013. Randomized load control: A simple distributed approach for scheduling smart appliances. In *International Joint Conference on Artificial Intelligence (IJCAI)*.
- Varaiya, P. P.; Wu, F. F.; and Bialek, J. W. 2011. Smart operation of smart grid: Risk-limiting dispatch. *Proceedings of the IEEE* 99(1):40–57.