Smart Device Power Allocation Via Swarm Intelligence

Jeremy Morton* and Mark Paluta*
Stanford University, Stanford, CA, 94305

This paper investigates task scheduling for smart devices when power constraints are of concern. The problem is formulated as a Resource-Constrained Project Scheduling Problem (RCPSP), where the tasks must be completed as quickly as possible subject to resource constraints. To solve this problem, Ant Colony Optimization (ACO), a form a swarm intelligence, is used to generate a schedule for device power usage. The ACO technique is compared to manual allocation techniques for several classes of problems within this RCPSP structure. ACO proves to be a superior method in problems where resource constraints are strict relative to the demands of individual devices, while manual techniques prove nearly as effective in problems without strict power constraints.

I. Introduction

In the modern power grid, devices are run at the behest of users with limited regard to global power usage, causing surges in power consumption during the afternoon and dips overnight. In the future, as new devices (i.e. electric vehicles) are introduced, the power grid may not be able to satisfy user needs immediately. This problem could be exacerbated in off-grid residential areas with limited power generation resources, where activating devices as soon as a task arises could cause the neighborhood to exceed its available power budget. Such an environment would require a strategic allocation of device tasks throughout the day to remain below its available power level.

The problem of task allocation subject to power constraints can be modeled as a Resource-Constrained Project Scheduling Problem. The RCPSP is a general scheduling problem that has been studied extensively using multiple optimization methods, such as genetic algorithms and simulated annealing.[1,2] It has been shown by Merkle *et al.*[3] that Ant Colony Optimization outperforms its counterparts in solving the RCPSP. Ant Colony Optimization, originally proposed by Dorigo,[4] generates project schedules through simulating the behavior of ants in search of food. It has been used in the past for task assignment[5] and multi-robot cooperation[6,7] problems, making it well suited for this project.

In this paper, an algorithm for solving the device task allocation problem using Ant Colony Optimization is developed. Its performance is then validated on a small problem, and extended to larger, more complex scenarios. Finally, the performance of ACO relative to manual device allocation methods is evaluated.

II. Problem Formulation

A. Resource-Constrained Project Scheduling Problem[3]

The Resource-Constrained Project Scheduling Problem involves the scheduling of activities in a project such that the timespan of the entire project is minimized. In solving the problem, resource and precedence constraints must be observed. In the traditional problem statement, the set of all activities is denoted by J. Each activity $j \in J$ has an associated duration, d_j, and resource requirement, r_j. In addition, every activity has a set of predecessors, P_j, and a set of successors, S_j, as illustrated in Figure 1.

The schedule for a project is represented by the start time of each activity, s_j, and the finish time of each activity, $f_j = s_j + d_j$. The timespan of the entire project is the difference between the latest finish time of any activity and the earliest start time. A schedule is deemed feasible if 1) each activity $j \in J$ does not start before all of its predecessors are finished and 2) for every time unit t, the sum of the activity resource requirements does not exceed the resource capacity at that time.

*M.S. candidate, Department of Aeronautics and Astronautics, Stanford University

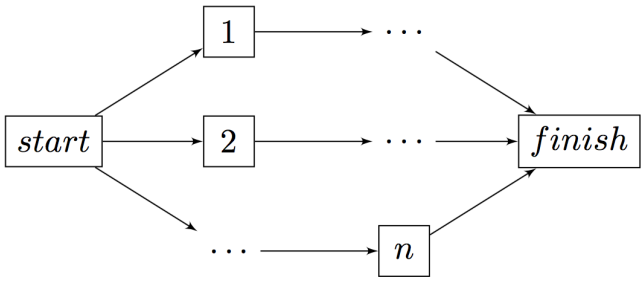


Figure 1. Illustration of precedence relationship between devices. Each activity is preceded and succeeded by another activity. Activities at the front of the chain are preceded by the start, while activities at the end of the chain are succeeded by the finish.

In extending this to the problem of power allocation amongst devices, the objective is to generate a power usage schedule for each device. The resource requirement of each device is the amount of power it consumes, and the overall resource capacity is the amount of available power at each time step. The precedence constraints can be viewed as enforcements that two devices cannot concurrently draw power. This could be desired for sets of devices such as washers/dryers that must run in a certain order, or in scenarios where households are restricted from running multiple devices at once to ensure the available power is distributed evenly among all residents. Thus, the problem consists of finding a power allocation schedule for a set of devices such that power and precedence constraints are met, and the overall timespan is minimized.

B. Ant Colony Optimization Algorithm[3]

In order to solve this problem, the Ant Colony Optimization approach will be used. In ACO, generations of artificial ants generate solutions through making a sequence of probabilistic decisions. The way in which a solution is reached is modeled based on the food-seeking behavior of ants in the wild. As ants search for food, they leave a trail of pheromones behind them that can be sensed by other ants. If a given path turns out to be fruitful, other ants will tend to follow same path, and the pheromone scent will grow stronger over time. If, on the other hand, a path is unrewarding, the pheromone scent will evaporate over time and other ants will tend to avoid that path. For the smart device power allocation problem, the probability of scheduling activity j during time step i is given by:

$$p_{ij} = \frac{(\sum_{k=1}^{i} \tau_{kj})^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h\in\epsilon}(\sum_{k=1}^{i} \tau_{kj})^{\alpha}[\eta_{ih}]^{\beta}} \tag{1}$$

where τ_{ij} is the pheromone associated with assigning activity j at time step i. The summation of pheromone levels over previous time steps allows for the selection of activities that had high pheromone levels at previous time steps but were not assigned at one of those times. The quantity η_{ij} is a problem-dependent heuristic, in this case given by:

$$\eta_{ij} = d_j + \sum_{i\in S_j} d_i - \min_{k\in\epsilon}(d_k + \sum_{i\in S_k} d_i) + 1 \tag{2}$$

where ϵ is the set of eligible devices at the given time step. The value of this heuristic biases the device selection toward devices with long collective durations between them and their successors. The values α and β in Eq. (1) determine the relative influence of the pheromone and heuristic values. After each generation of ants forms a set of solutions, the pheromone values are altered according to an evaporation and updating procedure. The pheromone evaporation is given by:

$$\tau_{ij} = (1-\rho)\tau_{ij} \tag{3}$$

where ρ is a quantity between 0 and 1. The pheromone updating is performed according to Eq. 4:

$$\tau_{ij} = \tau_{ij} + \rho \cdot \frac{1}{2T^*} \tag{4}$$

where T^* is the timespan associated with the best solution that has been found. Thus, this process leads to an increase in pheromone values associated with device assignments that have led to the best outcomes. After this pheromone update, a new generation of ants searches for a solution, and this process continues until an iteration limit is reached. A general outline of the algorithm can be found in Algorithm 1.

```
Generate devices
for Number of iterations do
    for Each ant do
        for Each time step, g do
            Generate list of eligible devices, ϵ(t_g)
            while ϵ(t_g) not empty do
                Calculate probability of choosing each device
                Choose a device based on probabilities
                Update power curve and ϵ(t_g)
            end
        end
        Find timespan, t
        Update t_local and t_global
    end
    Update pheromones, τ_ij
end
```

Algorithm 1: Ant Colony Optimization Algorithm

C. Device Modeling and Power Curve Generation

Each device was given a constant, random power level from a uniform distribution between 1 and 10 kW. The duration of each task was randomly assigned from a uniform distribution within different time ranges, with the probability of falling within the different ranges shown in Table 1. Additionally, to model the resource

Table 1. Probability distribution for device durations.

Duration (Hours)	**1 - 2**	**2 - 3**	**3 - 4**	**4 - 5**	**5 - 6**
Probability	0.35	0.25	0.2	0.1	0.1

constraints, a baseline power curve and maximum allowable power curve were generated. The baseline curve represents the power consumption of non-smart devices and was modeled as a sine wave centered at 10:30 AM. The maximum allowable power curve was based off of the concept of an off-grid residential area, which would be largely reliant on solar power and battery storage. Thus, the curve has a nominal battery level during the night and a periodic increase during daytime hours. Some liberty was taken in setting the mean power levels and amplitudes in order to accommodate the testing of power allocation among a large number of devices, and to allow the allocation to finish over a reasonable time frame. These curves are plotted in Figure 2.

III. Results

A. Validation

In order to determine if the algorithm had been implemented correctly, it was tested on the simple problem depicted in Figure 3(a). Using generations consisting of five artificial ants, the optimal solution was found within five iterations. This is shown in Figure 3(b).

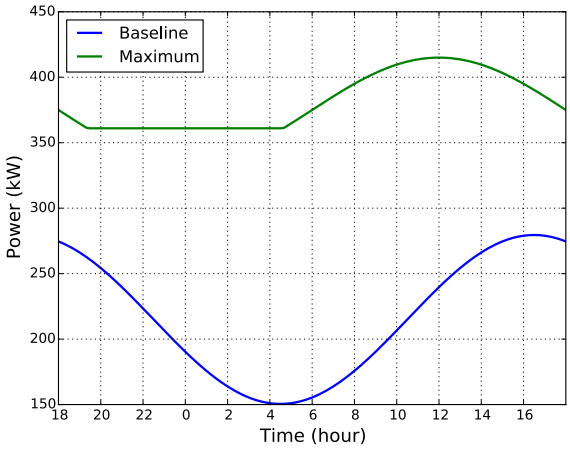
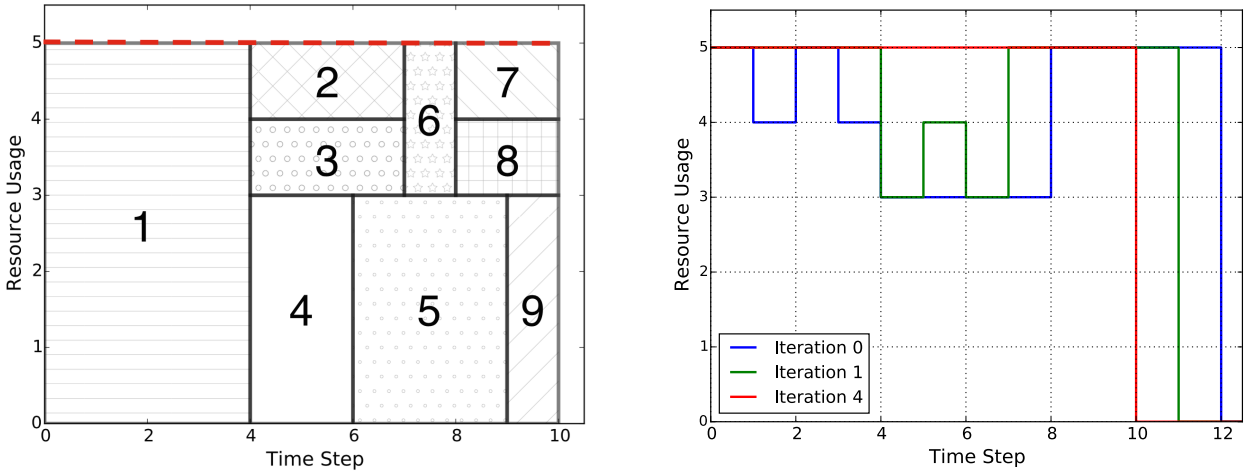


Figure 2. Baseline and maximum power curves used in simulations.



(a) Problem Illustration

(b) Simulation Results

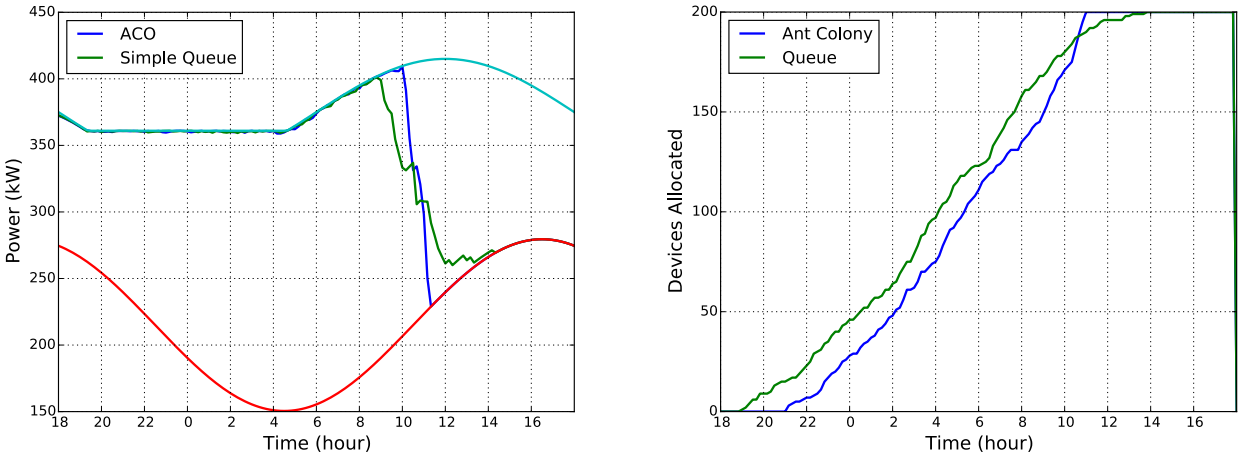
Figure 3. Validation of ACO algorithm on simple device allocation problem.

B. Comparison – Manual Device Allocation

1. Simple Queue

In the first test case, 200 devices are generated without any precedence constraints. In order to test the relative effectiveness of the ACO algorithm, a manual placement method was conceived. This method, the Simple Queue, begins at the first time step and iterates through the devices in the order they were generated, activating them until no more can be turned on without exceeding the maximum allotted power. It then iterates through time steps, activating as many devices as possible that have not yet completed their task, but always moving through the list of devices in the originally generated order. ACO outperforms the Simple Queue, as shown in Figure 4(a).

In Figure 4(b), which illustrates the number of devices whose task is complete at each time step, one can see a somewhat counterintuitive result. While ACO tends to finish allocating devices significantly sooner than the Simple Queue, it actually allocates devices more slowly through the majority of the simulation. This leads to an important insight – that the improved performance from ACO may be largely attributable to preferentially allocating devices with longer durations early on in the simulation.



(a) Comparison

(b) Device Allocation

Figure 4. Comparison of performance of Ant Colony Optimization and simple queue.

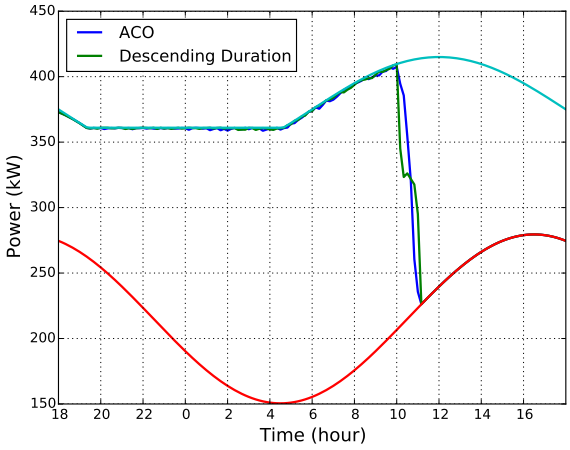


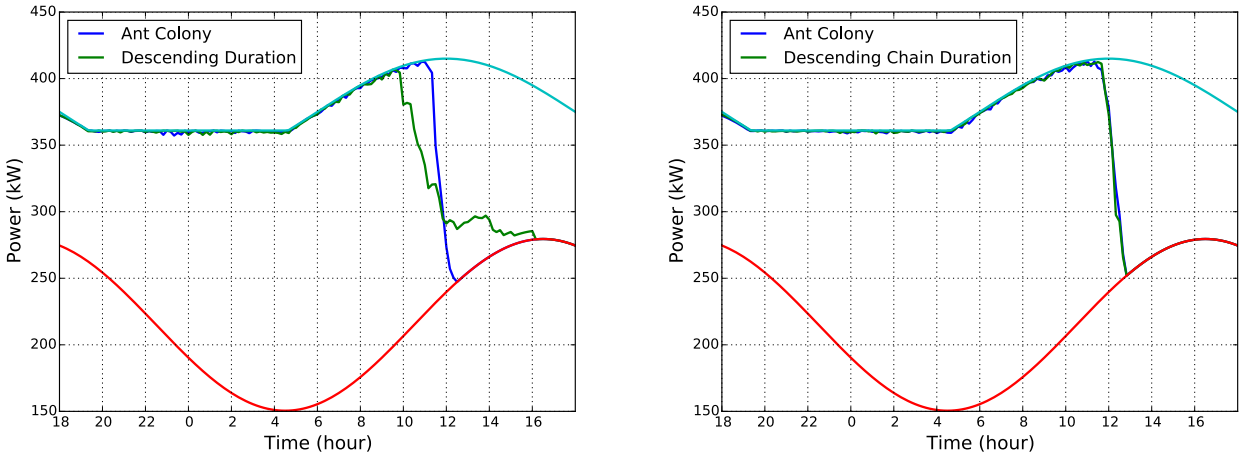
Figure 5. Performance of Descending Duration sorting scheme relative to ACO.

2. Queue Sorting and Precedence Constraints

The second manual technique employed builds upon this observation. As made apparent in the previous test, activating longer-duration devices near the start is generally advantageous for a shorter global allocation time. In order to mimic this behavior, a new type of queue was generated, which first sorts devices in descending order by duration. It then allocates device power allocation based on the same queueing procedure as the Simple Queue. This technique, deemed Descending Duration, leads to greatly improved performance, as shown in Figure 5.

Next, devices were placed in two-element chains, where precedence constraints were enforced. This constraint reduced the effectiveness of the Descending Duration method, as shown in Figure 6(a). The decrease in performance can be attributed to the fact that this method allocates devices without considering the effect on the other device in the chain.

In order to account for this constraint, a third manual technique, Descending Chain Duration, was conceived. This method sorts the devices by total chain duration, rather than their individual duration. As shown in Figure 6(b), this technique achieves largely the same performance as ACO.



(a) Descending Duration

(b) Descending Chain Duration

Figure 6. Performance of (a) Descending Duration and (b) Descending Chain Duration queue sorting schemes with precedence constraints enforced.

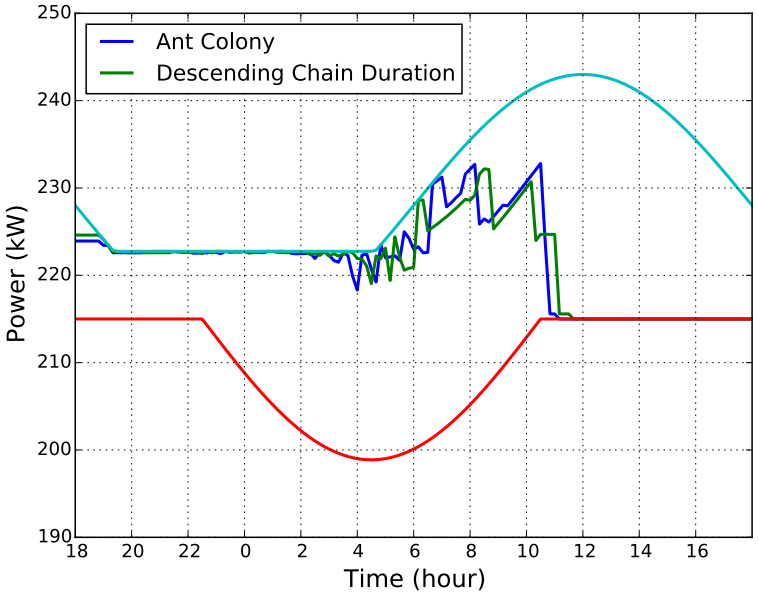


Figure 7. Performance of Descending Chain Duration sorting scheme relative to ACO with tight resource constraints.

3. Effect of Tightened Power Constraints

In the above scenarios, the resource capacity at any given time step is much larger in magnitude than any of the individual device resource requirements. In these cases, it is relatively easy to place devices such that the power usage is very close to the maximum power. This is illustrated in the previous figures, where the device power curves are nearly identical to the maximum power curve for much of the simulation. In this sense, not much strategic planning is needed in order to avoid wasting the available resources. Thus, it may be the case that the full advantage of using ACO can only be realized on a more tightly constrained problem.

By modifying the baseline curve to be flat in the afternoon and evening, the baseline and maximum curves could be moved much closer together. Additionally, 90% of the devices were reduced to a tenth of their original power level, creating more disparity in device power levels. In addition, the number of devices was halved so that the allocation could be completed over a comparable time frame to the previous problems. As

a result, the resource capacity was scaled down to a similar order of magnitude as the resource requirements of some devices. This modification leads to improved performance by ACO relative to Descending Chain Duration, as shown in Figure 7.

C. Results Summary

Table 2 summarizes the results of implementing ACO and the manual methods on each of the three aforementioned problems: the standard device allocation problem, the addition of precedence, and the enforecement of strict resource constraints.

These results are the mean improvement of ACO relative to the respective manual allocation methods over 100 simulations. Each simulation allowed ACO 15 iterations except for *With Precendence* against Descending Chain Duration, in which ACO was given 100 iterations. In a prior simulation where ACO was allowed only 15 iterations, Descending Chain Duration showed slightly superior performance.

Table 2. Comparison of allocation times relative to ACO (mean over 100 simulations, in minutes)

Allocation Method	Standard	With Precedence	Strict Resource Constraints
Simple Queue	+ 154.2	+ 265.3	+ 123.9
Descending Duration	+ 6.5	+ 192.2	+ 85.4
Descending Chain Duration	-	+ 2.1	+ 33.2

IV. Conclusion

This study has shown that Ant Colony Optimization is an effective method for solving the Resource-Constrained Project Scheduling Problem extended to smart device power allocation. While ACO shows superior performance in all cases, its benefit is trivial in loosely-constrained problems. In these problems, a manual method based on prioritizing long-duration tasks is nearly as effective. The benefit of ACO is more fully realized in tightly-constrained problems, where it tends to be more efficient in using the available power resources.

Work Distribution

For this project, Jeremy developed the algorithm for device allocation, while Mark developed the models for the devices and power curves. Both authors were involved in running simulations and writing this paper.

References

[1]Hartmann, S., "Self-adapting genetic algorithm with an application to project scheduling," Univ. of Jiel, Kiel, Germany, Tech. Rep. 506, 1999.

[2]Bouleimen, K. and Lecocq, H., "A new efficient simulated annealing algorithm for the resource constrained project scheduling problem," *Service de Robotique et Automatization*, Univ. de Lige, Lige, Belgium, 1998.

[3]Merkle, D., Middendorf, M. and Schmeck, H., "Ant colony optimization for resource-constrained project scheduling," *IEEE Trans. Evol. Comput.* **6**(4) (2002), pp. 333-346.

[4]Dorigo, M., M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comp. Intell. Mag.,* Vol. 1, No. 4, pp. 28-39, Nov. 2006.

[5]Zhang, D., *et. al.*, "An adaptive task assignment method for multiple mobile robots via swarm intelligence approach," *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 05)*, pp. 415-420, June 2005.

[6]Ding, Y., He, Y., and Jiang, J., "Multi-robot cooperation method based on the ant algorithm," *Swarm Intelligence Symposium (SIS 03)*, 2003.

[7]Xu, S., *et al.*. "Ant-based swarm algorithm for charging coordination of electric vehicle," *International Journal of Distributed Sensor Networks*, Apr. 2013.