Smart Grid Power Scheduling via Bottom Left Decreasing Height Packing

Anshu Ranjan, Pramod Khargonekar, Sartaj Sahni
University of Florida
aranjan@cise.ufl.edu, ppk@ece.ufl.edu, sahni@cise.ufl.edu

***Abstract*—We consider the scheduling of flexible electric power loads so as to minimize the peak load. We settle a conjecture of Alamdari et al.[1] regarding the performance of the bottom left decreasing height heuristic to schedule preemptable loads with known power requirement and duration and having the same earliest start time and the same deadline. Specifically, we show a tight bound of $4/3 - 1/(3D)$ relative to the minimum peak power load, where D is the duration of the scheduling interval. On benchmark strip packing data, the bottom left decreasing height heuristic generated schedules that required up to 45% less peak power than required by schedules generated using the next fit decreasing height heuristic. On electric and plug-in hybrid electric vehicles data, these two heuristics are very competitive.**

I. Introduction

Electric power systems are in the process of significant transformation. These changes are driven by the desire to integrate greater amounts of renewable electric energy from wind and solar generation. Unlike traditional electric power generation technologies such as coal, gas, nuclear, and hydro, wind and solar energy is inherently uncertain, variable, and (largely) uncontrollable. These characteristics pose major challenges in power systems operations. In particular, the requirement to balance the generation and consumption on a second-by-second basis becomes much more difficult at high levels of penetration of renewable electricity. At the same time, population growth, economic growth, and new modes of electric energy consumption (e.g., electric cars) are expected to increase demand for electric energy. Power system infrastructure (generation, transmission, distribution) is driven in large part by the peak demand and therefore it is important to reduce the increase in peak power demand even as electric energy consumption increases.

The traditional paradigm for power systems operations is to adjust supply to meet variable demand. In the context of deep penetration of renewable energy, this paradigm requires increasing amounts of reserve generation capacity [2]. In recent years, there is considerable interest in exploiting the inherent flexibility in certain types of loads. For example, dishwashers, clothes washers, dryers, water heaters, HVAC units, and electric vehicles can meet the end user needs without a rigid power consumption schedule. Specifically, an electric vehicle charging load can meet user requirement so long as the car is charged by a certain deadline. By exploiting the flexibility of large numbers of flexible loads, it may be possible to adjust demand and integrate larger amounts of renewable generation without large increases in reserves. At the same time, it is possible to reduce the increase in peak power demand by exploitation of flexible loads.

There are two key problems in benefiical utilization of flexibility in loads. One is to ex-ante aggregate flexibility in large number of loads for benefit to the operation of the power system via various electricity markets and operational procedures. The second is to deliver a required amount of flexibility at run-time by suitable scheduling of flexible loads. This paper considers a specific problem along the former direction. In particular, we explore the problem of minimizing peak power demand by scheduling of flexible loads. As in [3] and [4] we limit ourselves to the offline cost-optimal scheduling problem (OCOSP) in which all jobs have the same earliest start time and the same deadline and in which the power and duration of each job are known in advance. We focus on a specific heuristic called "bottom left decreasing height preemptive" $BLDHPreemptive$ and obtain an upper bound on the ratio of the peak power required by a schedule constructed using this heuristic and that of the offline preemptive cost optimal scheduling policy. We show that this ratio cannot exceed $4/3 - 1/(3D)$ where D is the duration of the scheduling interval. This settles a conjecture of Alamdari et al. [1] regarding the worst-case performance of $BLDHPreemptive$ (in [1], [5], $BLDHPreemptive$ is referred to as First-Fit Decreasing). The performance of $BLDHPreemptive$ is compared experimentally with that of the preemptive version of another simple scheduling heuristic–next fit decreasing height.

II. Related Work

The relationship between strip packing and preemptive OCOSP has been studied by Alamdari et al. [1] and [5]. They prove that preemptive OCOSP is NP-hard by reduction from the partition problem[5]. They have proposed a fully polynomial time approximation scheme which involves the use of linear programs. The FPTAS is mainly for theoretical interest, as the authors have indicated, because of their running time and no limit on number of preemptions per job. Additionally, they propose an $O(n^2)$ algorithm with a performance ratio of 3/2, an $O(n \log(nM))$ algorithm with a performance ratio of 5/3 (this algorithm has at most 3 preemptions per job), and an $O(n \log^2 n \log(nM)/ \log\log n)$ algorithm with performance ratio 5/3 that has at most 1 preemption per job. They conjectured the bound of the heuristic discussed in this paper, $BLDHPreemptive$, to be 4/3 and in [5] an example for which the ratio is arbitrarily close to 4/3 is provided but the bound is not established. We settle this conjecture by proving a better worst-case performance bound (4/3-1/($3D$)), which is better than all but the FPTAS of [1], [5]. There can be at most n preemptions per job in this heuristic and can be easily implemented in $O(n^2)$ time (Pseudo code in Section III) by maintaining a list of the intervals sorted by the total power

scheduled in them.

In Section III, we describe the $BLDHPreemptive$ heuristic. The performance bound is proved in Section IV. Section VI is the Conclusion.

III. $BLDHPreemptive$ Heuristic

$BLDHPremptive$ (Pseudo code shown below and an illustrative example in Figure 1) is an adaptation of the bottom left decreasing height non-preemptive heuristic of strip packing [6] for preemptive OCOSP. In this heuristic, n is the number of jobs in an preemptive OCOSP instance; d_i and p_i, respectively, are the duration and power requirement of the ith job; and D is the deadline by which all jobs must complete. The d_is, p_is, and D are positive integers greater than 0.

Algorithm 1 $BLDHPreemptive$

Input: A 2D array of integers, $J[1,2,...,N][1,2]$: $J[i][1]$ and $J[i][2]$ indicate power and duration respectively of job i, an integer D: total duration.

Output: A list of mappings between job index, j i.e. an integer indicating the index of job, and list of intervals ($j \rightarrow \{intervals\}$). Each interval is defined by two integers: $(startT, endT)$ representing start time and end time respectively of the the time interval when the job j is to be scheduled.

1: Initialize a list, $schedule[1,2..N]$, of mappings between job index and list of intervals, $schedule$. Each job j maps to an empty list of intervals.
2: Initialize a list, $sortedL$, of an array of three integers, $sortedL[1..][1,2,3]$ with a single entry $(1, D, 0)$. The first two integers indicate start and end times of an interval and the third integer indicate the total power of jobs of scheduled in the interval.
3: Re-index jobs in J such that $J[i][1] \geq J[i+1][1]$ for $1 \leq i \leq N-1$. If $J[i][1] = J[i+1][1]$, then $J[i][2] < J[i+1][2]$.
4: **for** $j = 1$ to N **do**
5: $d = J[j][1]$ and $p = J[j][2]$
6: Initialize an empty list, $tempL$ similar to $sortedL$
7: **while** $(d > 0)$ **do**
8: Extract first entry, $sortedL[1][1,2,3]$, from $sortedL$ and assign it to $e[1,2,3]$
9: **if** $e[2] - e[1] > d$ **then**
10: Split e into two: $e1[1,2,3] = (e[1], e[2]-d, e[3]+p)$ and $e2[1,2,3] = (e[2]-d, e[2], e[3])$.
11: Add $e1$ to $tempL$ and $e2$ to $sortedL$ in **sorted** order of total power i.e. third index.
12: Add the interval $(e1[1], e1[2])$ to $schedule[j]$.
13: **else**
14: $e[1,2,3] = (e[1], e[2], e[3]+p)$
15: Add e to $tempL$ in **sorted** order of total power i.e. third index.
16: Add the interval $(e[1], e[2])$ to $schedule[j]$.
17: **end if**
18: **end while**
19: Extract all entries of $tempL$ and add them to $sortedL$ in sorted order of total power i.e. third index.
20: **end for**
21: **return** $schedule$

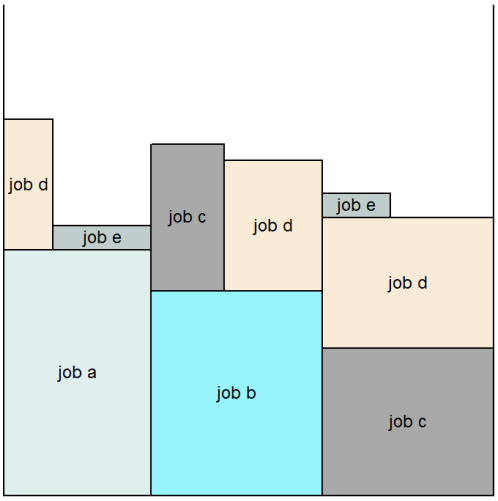


Fig. 1: $BLDHPreemptive$ schedule of jobs a, b, c, d and e with dimensions $(p \times d)$: 30×30, 25×35, 18×35, 16×65 and 14×40 respectively. The total duration, $D = 100$

In this heuristic, jobs are scheduled one at a time in decreasing order of height. Since all job duration are integer, all preemption are at integer times[1]. So, (partial) preemptive schedules generated by $BLDHPreemptive$ have the property that the total power committed[2] is constant within each unit time interval $j, 0 \leq j < D$ (the unit interval j represents the time from j to $j+1$). Let $power[j], 0 \leq j < D$ denote the power currently committed in the unit interval j. To schedule the job i currently being considered, we find the unit interval t that satisfies the following properties:

P1: Job i is not already scheduled in the interval.
P2: $power[t]$ is minimum from among the intervals that satisfy P1. That is, t is a unit interval with least committed power and in which job i is not already scheduled.

In case of a tie, we pick the smallest such t (hence the name bottom left). Let this t be denoted by $tStart$. Let $tFinish$ be the largest t such that $power[tStart] = power[t]$, $tStart \leq t < tFinish$ and job i is not already scheduled at any time between $tStart$ and $tFinish$. That is, $tStart$ to $tFinish$ is the longest interval that begins at $tStart$, has a committed power of $power[tStart]$ during this interval, and job i is not already scheduled anywhere in this interval. $BLDHPremptive$ schedules as much of the job i currently being scheduled as is possible in the leftmost portion of this interval. In case some of job i remains to be scheduled, the remainder is scheduled using the same strategy iteratively. When all of job i has been scheduled, we advance to the next job in decreasing height order. Note that when job i (or part of job i) is scheduled, the power committed increases by p_i in the newly scheduled unit intervals.

[1] Alamdari et al. [5] have demonstrated an instance of preemptive OCOSP in which all d_i are 1, D is an integer and the optimal schedule has preemption at non integer times. This instance has a 1% lower peak power requirement than any schedule that preempts only at integer times.

[2] The power committed at time t is the sum of the power requirements of jobs scheduled at time t.

IV. Performance Bound

The proof is based on the proof of Longest Processing Time algorithm by Graham [7].

A. Notations

Without loss of generality, we assume that the jobs are indexed so that $p_1 \geq p_2 \geq \cdots \geq p_n$ and that $BLDHPreemptive$ schedules the jobs of I in this order. Let $OPT(pre)$ be the peak power demand of an optimal preemptive schedule for I and let $BLDH(pre)$ be that of the schedule generated by $BLDHPreemptive$.

We divide the time line $[0, D]$ of both $OPT(pre)$ and $BLDH(pre)$ into **slices** with the property that there is no preemption within a slice. Since the optimal schedule has a finite number of preemptions, the number of slices is also finite.

Consider any slice of any schedule a $BLDHPreemptive$ schedule. The job scheduled in this slice with the least index is called the **bottom** job. In case at most two jobs are scheduled in this slice, we call the other job (if any) the **top** job.

Consider the (partial) $BLDHPreemptive$ schedule immediately after job i of I has been scheduled. We say that job i has white space below it if, at this time, there is a slice s_j in which job i has been scheduled and another interval s_q such that $power[j] - p_i > power[q]$. In the $BLDHPreemptive$ schedule of Figure 1, job c and e has white space below them (immediately after they were scheduled). Let W be the subset of jobs of I that have white space below them.

B. The Bound

We establish the following theorems:

Theorem 1: When $p_n > OPT(pre)/3$, the schedule generated by $BLDHPreemptive$ is optimal.

Proof: This follows from Lemma 1 to Lemma 3. ■

and

Theorem 2: For every instance of preemptive OCOSP,

1)
$$\frac{BLDH(pre)}{OPT(pre)} \leq \frac{4}{3} - \frac{1}{(3D)}$$
where D is the schedule deadline;

2) this bound is tight.

For Theorem 2, we first observe that when $D = 1$, $d_i = 1$ (recall that all d_i are required to be integer and in the range $0 < d_i \leq D$) for all jobs and all jobs are scheduled from 0 to 1 by $BLDHPreemptive$. It is easy to see that the $BLDHPreemptive$ schedule is optimal and so satisfies the upper bound of the theorem. In the following, we prove the upper bound for $D > 1$. The tightness of the upper bound (i. e. Theorem 2[ii]) is established by the example in [5].

When $D > 1$, the upper bound of Theorem 2(i) is established by contradiction. Let I be a smallest (i.e., has the least number n of jobs and among instances with the same number of jobs has the smallest value of $\sum p_i d_i$) instance that violates the upper bound. [3] Note that since I is the smallest instance that violates the bound, the peak power demand of the $BLDHPreempitive$ schedule for jobs 1 through $n-1$ of I is less than $BLDH(pre)$ as otherwise the first $n-1$ jobs define a smaller instance than I that violates the bound.

We consider two cases depending on whether or not job n has white space below it. The proof for the case when job n has white space below it can be found in [4]. For the rest of the section we assume that the last job of I does not have white space below it. We first prove Theorem 1. As per the statement of the theorem, we assume that $p_n > OPT(pre)/3$.

Note that for schedules generated by $BLDHPremptive$, the first job assigned to an interval is the bottom job and the indexes of the bottom jobs for I are 1 for times in the intervals 0 through $d_1 - 1$ and 2 for times in the intervals d_1 through $\min\{d_1 + d_2, D\} - 1$.

Lemma 1: There is an optimal schedule for I in which the indexes of the bottom jobs are the same as those in the $BLDHPremptive$ schedule for every time instance in $[0, D]$.

Proof: Consider an optimal schedule for I. From [5], we know that this optimal schedule may have preemption at non-integer times. Divide the time line $[0, D]$ into slices with the property that there is no preemption within a slice in the optimal schedule. Since the optimal schedule has a finite number of preemption, the number of slices is also finite. Rearrange the slices of the optimal schedule so that the bottom job indexes are in non-decreasing order left to right. Let O be the result. Let B be the $BLDHPremptive$ schedule. Let a be the least time at which the bottom jobs have a different index in O and B. If there is no such a, the lemma is proved. So, assume that a exists. From the definition of a and the way $BLDHPreemptive$ works (see Figure 2), it follows that a is the start of a slice of O.

Let b_a and t_a, respectively, be the bottom and top jobs in O in the slice that begins at a and and let q be the bottom job scheduled at time a in B. From the ordering of the slices in O and the nature of the $BLDHPremptive$ heuristic, it follows that $q < b_a < t_a$. Also from the ordering of slices in O, it follows that there is a slice of O that begins at time c, $c < a$ and has q as a top job. Let b_c be the bottom job in this slice of O. It follows that $b_c < q < b_a < t_a$. Let Δa and Δc be the durations of slices a and c. So, swapping $\delta = \min\{\Delta a, \Delta c\}$ of q and b_a in O (see Figure 3) results in a schedule O' in which no interval has the same job scheduled twice (once as a bottom job and once as a top job) and in which the bottom jobs agree with those of B from time 0 to time $a+\delta$. Further, since $q < b_a$ and $b_c < t_a$, $p_{b_a} + p_{b_c} \leq p_q + p_{b_c}$ and $p_q + p_{t_a} \leq p_q + p_{b_c}$. So, the swap does not increase the maximum power committed and O' is also an optimal schedule.

We repeat the above transformation as many times as needed to arrive at an optimal schedule in which the bottom jobs agree with those in B. To see that the number of needed transformations is finite, note that when $\Delta c < \Delta a$, the above transformation results in an O' that has one fewer slice than

[3] Since the p_is and d_is are integers, $\sum(p_i * d_i)$ is also an integer and so among the instances that violates the bound, there must be an instance with minimum $\sum(p_i * d_i)$ i.e. there cannot be an infinite sequence of instances with strictly decreasing $\sum(p_i * d_i)$.

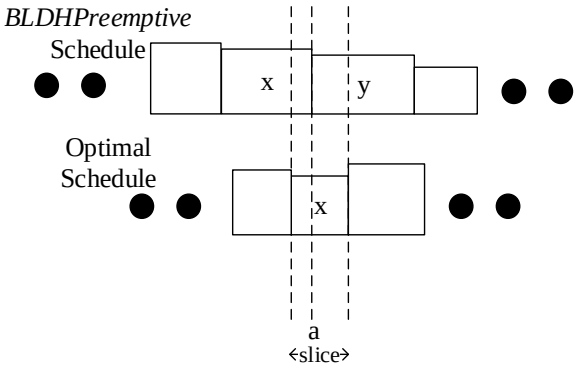


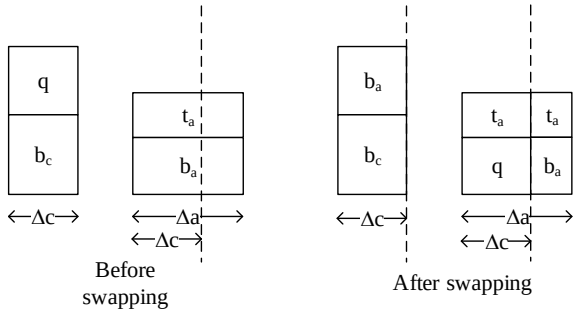
Fig. 2: An impossible case when the least time, a, at which the bottom jobs have a different index in the BLDH$Preemptive$, B, and the optimal schedule, O. It is impossible because the since both B and O have agreed until a, both must have scheduled the job x totally before a. Note that BLDH$Preemptive$ schedules the job till completion without preemption unless it reaches the width of the strip.

O in which q is a top job and has one additional slice to the right of a (the new slice begins at $a + \delta$); when $\Delta c = \Delta a$, O' has one fewer slice in which q is a top job and the same number of slices to the right of a; and when $\Delta c > \Delta a$, O' has as many slices as O in which q is a top job and has one additional slice of the left of a (see Figure 3). In all cases, O' and B agree in their bottom jobs up to time $a + \delta$. The last two cases reduce the number of slices in which the optimal schedule and B may disagree in their bottom jobs by 1 while the first case may not reduce in the number of differing slices. However, since the number of slices to the left of a that have q as a top job is finite, the first case can be done only a finite number of consecutive times before only one slice has q as a top job. At this time, we must fall into one of the other two cases and reduce the number of slices in which the (current) optimal schedule and B disagree in their bottom jobs by 1. So, the stated transformation is done only a finite number of times and hence must terminate with an optimal schedule that satisfies the lemma. ■

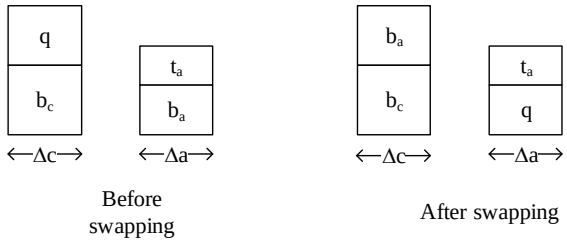
Lemma 2: There are at most two jobs scheduled at any time in the *BLDHPreemptive* schedule of I.

Proof: Consider an optimal schedule O which agrees with the $BLDHPremptive$ schedule B in its bottom jobs as per Lemma 1. Let DT be the sum of the durations of the slices of O that have a top job. Let DB be the sum of the durations of the slices of O (and hence also of B) that have a bottom job with a power requirement that is less than $2OPT(pre)/3$. From the proof of Lemma 4, we know that $p_n > OPT(pre)/3$. So, $DT \leq DB$.

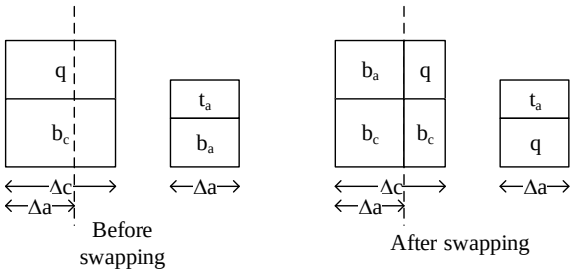
Suppose that there are 3 jobs scheduled in some unit interval j of B (recall that $BLDHPreemptive$ may preempt only at integer times). At the time the third (say q) job was assigned to this interval, the power committed in this interval was at least $2p_n > 2OPT(pre)/3$ (because each of the two already scheduled jobs in this interval have a power requirement that



(a) $\Delta c < \Delta a$



(b) $\Delta c = \Delta a$



(c) $\Delta c > \Delta a$

Fig. 3: Different cases of swapping described in Lemma 1

.

is at least p_n). From the way $BLDHPreemptive$ works, it follows that at this time, the power committed in B at every time in the interval $[0, D]$ was $> 2OPT(pre)/3$ and so all bottom jobs having a power requirement $< 2OPT(pre)/3$ have a second job assigned. So, the duration, in B, of already assigned second jobs is $\geq DB$. Adding in the duration of q, we see that the total duration of jobs other than bottom jobs is more than DB, which contradicts $DT \leq DB$. So, in B, there is no time at which 3 or more jobs are scheduled. ■

Lemma 3: There is an optimal schedule for I that is identical to the $BLDHPreemptive$ schedule B for I.

Proof: Please refer to [4] for the proof. ■

This proves Theorem 1. We now prove Theorem 2 for the case when p_n does not have white space below it.

Lemma 4: With the assumption that the last job n of I

does not have white space below it, $p_n > OPT(pre)/3$.

Proof: It is easy to see that

$$OPT(pre) \geq \frac{1}{D}\sum_{i=1}^{n} p_i d_i$$

From the nature of $BLDHPreemptive$, it follows that

$$\sum_{i=1}^{n-1} p_i d_i \geq (BLDH(pre) - p_n) * D$$

Using these two inequalities, we obtain

$$\begin{aligned}
&\frac{BLDH(pre)}{OPT(pre)} \\
&= \frac{BLDH(pre) - p_n + p_n}{OPT(pre)} \\
&\leq p_n/OPT(pre) + \frac{1}{(D.OPT(pre))}\sum_{i=1}^{n-1} p_i.d_i \\
&= p_n/OPT(pre) + \frac{1}{(D.OPT(pre))}\sum_{i=1}^{n} p_i.d_i \\
&- \frac{p_n.d_n}{D.OPT(pre)} \\
&= \frac{D.p_n - p_n.d_n}{D.OPT(pre)} + \frac{1}{(D.OPT(pre))}\sum_{i=1}^{n} p_i.d_i \\
&\leq \frac{p_n(D-d_n)}{D.OPT(pre)} + 1
\end{aligned}$$

Since I violates the bound of Theorem 2 and since $d_n \geq 1$,

$$\begin{aligned}
\frac{p_n(D-d_n)}{D.OPT(pre)} + 1 &> 4/3 - \frac{1}{3D} \\
\frac{p_n(D-d_n)}{D.OPT(pre)} &> \frac{(D-1)}{3D} \\
p_n/OPT(pre) > \frac{D-1}{3(D-d_n)} \quad &\geq \frac{D-1}{3(D-1)} = 1/3
\end{aligned}$$

■

Theorem 2 is now established for I using Lemma 4 and Theorem 1. The rest of the proof for Theorem 2 can be found in [4].

C. Implementation and Complexity of PreemptiveBLDH

$BLDHPreemptive$ may be implemented so at to run in $O(n^2)$ time[4], [5]. Since there are n-job instances for which $BLDHPreemptive$ generates $O(n^2)$ preemption (see Figure 4), our implementation is asymptotically optimal.

V. Experimental Evaluation

We compared the performance of $BLDHPreemptive$ relative to $NFDHPreemptive$ [3] algorithm for power scheduling. Section V-A gives the results using some of the strip packing benchmark datasets of Ntene et al. [8] and those used by us in [3]. Section V-B gives the results for datasets with only two power demands. The latter datasets simulate power demands for the charging of hybrid and electric vehicles. All experiments were performed on a PC with 3 GB memory and a dual core Intel E2180 2.00 GHz processor.

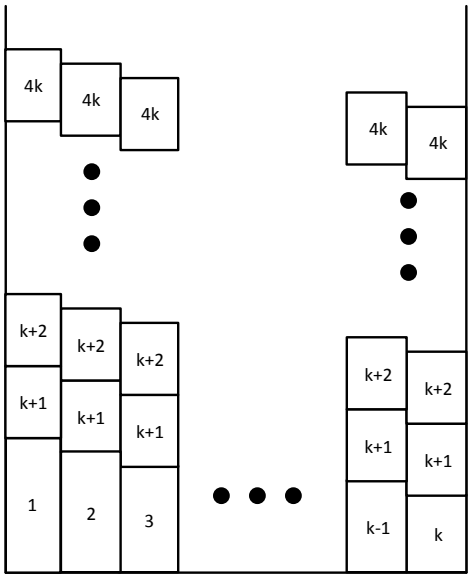


Fig. 4: Example showing $O(n^2)$ preemptions using $BLDHPreemptive$ algorithm. The jobs are labelled with its index (i for job J_i). There are n jobs. k = $n/4$ of these jobs ($J_1, J_2, ..., J_k$ are in decreasing order of power requirements) are scheduled at the bottom of the strip. The remaining $3n/4$ jobs each have duration D and same power requirement ($\leq$ power of J_k). Total number of preemption = $3k * k$ = $3n/4 * n/4$ = $O(n^2)$.

A. Datasets of [8]

We used those datasets used in [8] for which the minimum power required for each datasets is known, which were formed by repeatedly cutting a large rectangle. They are described below:

- *Mumford-Valenzuela Benchmark* - This dataset [9] was constructed by making guillotine cuts on a 100 x 100 rectangle. It comprises of two types of instances: *Nice* and *Path* with aspect ratios in the ranges $1/4 \leq Height/Width \leq 4$ and $1/100 \leq Height/Width \leq 100$, respectively. Also, the ratio of the areas of any two rectangles in a given dataset is between 7 and 100 respectively for *Nice* and *Path*.
- *Hopper and Turton Benchmark* - This dataset [10] consists of randomly generated rectangles with aspect ratio at most 7. The number of rectangles in each dataset is between 17 to 197.
- *Burke Benchmark* - Burke et al. [11] cut a large rectangle of different sizes repeatedly to create this dataset. The data is random restricted only by a minimum dimension allowed for a rectangle.

In all cases, the rectangles were mapped to jobs by equating the height and width of rectangles with power and duration,

respectively. Since the minimum power requirement is known for the datasets of [8], for these datasets, we computed the performance ratio, PR, which is equal to the ratio of power required by the schedule generated by one of the test algorithms and the optimal minimum power. The mean and standard deviation of values of PR were also calculated. We note that lower values of PR are better and $PR \geq 1$.

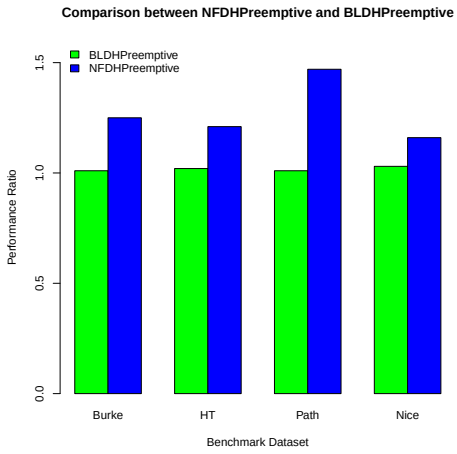


Fig. 5: Comparison between performances of $NFDHPreemptive$ and $BLDHPreemptive$ using benchmark data used in [8] (average over each dataset). Lower PR values implies better performance.

Figure 5 shows the plot for PR values for the datasets (average over each dataset) of [8] of the two heuristics. As can be seen, $BLDHPreemptive$ gives near optimal results with its average PR values very close to 1, the max being 1.05. The average PR ratios for $NFDHPreemptve$ are much higher achieving a high of 1.49. Furthermore, $BLDHPreemptive$ consistently performed better than $NFDHPreemptive$. The best IR (improvement ratio) relative to $NFDHPreemptive$ achieved on any instance was 1.81. For this instance, then, $BLDHPremtive$ achieved a peak power reduction of almost 45% relative to $NFDHPreemptive$.

B. Electric vehicles dataset

As in [3], we used a setup similar to that used by Pedro and Guillermo Sanchez [12] to simulate the charging of plug-in hybrid (PHEV) and electric vehicles (EV). The simulation comprised 6 sets of datasets each having 50 vehicles. Each set was characterized by the ratio of number of PHEV and EV - 0:100, 20:80, 40:60, 60:40, 80:20 and 100:0. The attributes of cars depended on their type: maximum battery charge were 23 kWh (EV) and 7.2 kWh (PHEV) and the charging power rate were 3.8 kW and 2.2 kW respectively. Similarly, we generated 5 sets of data with the same ratios of PHEV:EV with number of cars: $n \in 50, 100, 200, 500, 800, 1000$. Each set consisted of 5 instances for each of 6 ratios between number of PHEV and EV. Thus there were 180 instances in total. All the cars had the same starting time and deadlines to make it consistent with the problem that we studied. We set the total duration as 10 hours considering an overnight charging time (say 8 am to 6 pm). The maximum time needed to fully charge an PHEV and EV were set as 61 $(23/3.8 \times 10)$ and 33 $(7.2/2.2 \times 10)$ minutes respectively. The required charge time for each vehicle was randomly determined within the ranges 1 to 61 minutes for EVs and 1 to 33 for PHEVs.

The preemptive ILP of [3] run with a time limit of 150 seconds per instance, $BLDHPreemptive$, and $NFDHPreemptive$ result in schedules with almost identical peak power (the latter two take less than a tenth of a second per instance).

VI. Conclusion

We have settled a conjecture of Alamadari et al. [1] concerning the performance of $BLDHPreemptive$. Alamdari et al. conjectured a worst-case bound of 4/3. We have shown the bound is 4/3-1/(3D) and that this bound is tight. On benchmark strip packing data, the bottom left decreasing height heuristic generated schedules that required up to 45% less peak power than required by schedules generated using the preemptive next fit decreasing height heuristic. On electric and plug-in hybrid electric vehicles data, these two heuristics and the more compute intensive preemptive ILP of [3] are very competitive.

References

[1] S. Alamdari, T. Biedl, T. Chan, E. Grant, K. Jampani, S. Keshav, A. Lubiw, and V. Pathak, "Smart-grid electricity allocation via strip packing with slicing," in *Algorithms and Data Structures*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 8037, pp. 25–36.

[2] U. Helman, "Resource and transmission planning to achieve a 33% rps in california - iso modeling tools and planning framework," in *FERC Technical Conference on Planning Models and Software, 2010*, 2010.

[3] A. Ranjan, P. Khargonekar, and S. Sahni, "Offline preemptive scheduling of power demands to minimize peak power in smart grids," in *2014 IEEE Symposium on Computers and Communications (ISCC)*, June 2014, pp. 1–6.

[4] ——. Tight performance bound for bottom left decreasing height scheduling of preemptable power loads. [Online]. Available: do.http://www.cise.ufl.edu/~aranjan/Papers/BLDH.pdf

[5] S. Alamdari, T. Biedl, T. Chan, E. Grant, K. Jampani, S. Keshav, A. Lubiw, and V. Pathak. Strip packing with slicing. [Online]. Available: do.http://csclub.uwaterloo.ca/~egrant/ElyotResearchPapers/StripPackingwithSlicing.pdf

[6] B. Baker, E. Coffman, Jr., and R. Rivest, "Orthogonal packings in two dimensions," *SIAM Journal on Computing*, vol. 9, no. 4, pp. 846–855, 1980.

[7] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM Journal on applied mathematics*, vol. 17, no. 2, pp. 416–429, 1969.

[8] N. Ntene and J. H. Van Vuuren, "A survey and comparison of guillotine heuristics for the 2d oriented offline strip packing problem," *Discret. Optim.*, vol. 6, no. 2, pp. 174–188, May 2009.

[9] C. L. Mumford-Valenzuela, J. Vick, and P. Y. Wang, "Metaheuristics." Kluwer Academic Publishers, 2004, ch. Heuristics for Large Strip Packing Problems with Guillotine Patterns: An Empirical Study, pp. 501–522.

[10] E. Hopper and B. Turton, "An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem," *European Journal of Operational Research*, vol. 128, no. 1, pp. 34 – 57, 2001.

[11] E. K. Burke, R. S. R. Hellier, G. Kendall, and G. Whitwell, "A New Placement Heuristic for the Orthogonal Stock-Cutting Problem," *Operations Research*, vol. 52, no. 4, pp. 655–671, Jul. 2004.

[12] P. Sanchez-Martin and G. Sanchez, "Optimal electric vehicles consumption management at parking garages," in *PowerTech, 2011 IEEE Trondheim*, 2011, pp. 1–7.