RAIRO-Oper. Res. 52 (2018) 371–382
https://doi.org/10.1051/ro/2017035

RAIRO Operations Research
www.rairo-ro.org

RESOURCE ALLOCATION PROBLEM UNDER SINGLE RESOURCE ASSIGNMENT

Sakib A. Mondal[1,*]

Abstract. We consider a NP-hard resource allocation problem of allocating a set of resources to meet demands over a time period at the minimum cost. Each resource has a start time, finish time, availability and cost. The objective of the problem is to assign resources to meet the demands so that the overall cost is minimum. It is necessary that only one resource contributes to the demand of a slot. This constraint will be referred to as single resource assignment (SRA) constraint. We would refer to the problem as the S_RA problem. So far, only 16-approximation to this problem is known. In this paper, we propose an algorithm with approximation ratio of 12.

Mathematics Subject Classification. 68W25, 90B35, 68R01.

Received January 4, 2017. Accepted May 6, 2017.

1. Introduction

Energy sector is one of the most prominent industry today which is expected to grow enormously in future. One of the challenging tasks in the sector is to produce cost efficient energy. A firm producing electric power may have number of ways to generate power, namely from solar panels, wind mills, batteries and diesel generators. Most of the large power generating firms would have fairly good idea about the power demand profile based on past experiences. Cost of energy depends on the type of generating source. However quite often, it is optimal to use only one type of generator due to additional huge cost of overburden to maintain two generators operating simultaneously and feed their supplies efficiently. Therefore the goal of the power generator is to meet the demand profile incurring minimum cost while using only one type of sources during each time slot. Motivated by the problem of the power generator, we formulate the Single Resource Allocation (S_RA) problem.

1.1. The S_RA problem

The S_RA problem involves allocation of single or multiple copies of resources from a set of resources to complete a given set of jobs over a time horizon at minimum cost subject to the constraint that the demand of a slot be met by only one resource. The S_RA problem is a special case of the more general resource allocation (RA) problem [6, 11, 12]. The RA problem can be formally described as follows:

Keywords. Resource allocation, scheduling, primal dual.

[1] Data Science, Flipkart Internet Private Limited, Cessna Business Park Embassy Tech Square, Outer Ring Road, Bengaluru, Karnataka 560103.
*Corresponding author: sakib@hotmail.com

Article published by EDP Sciences

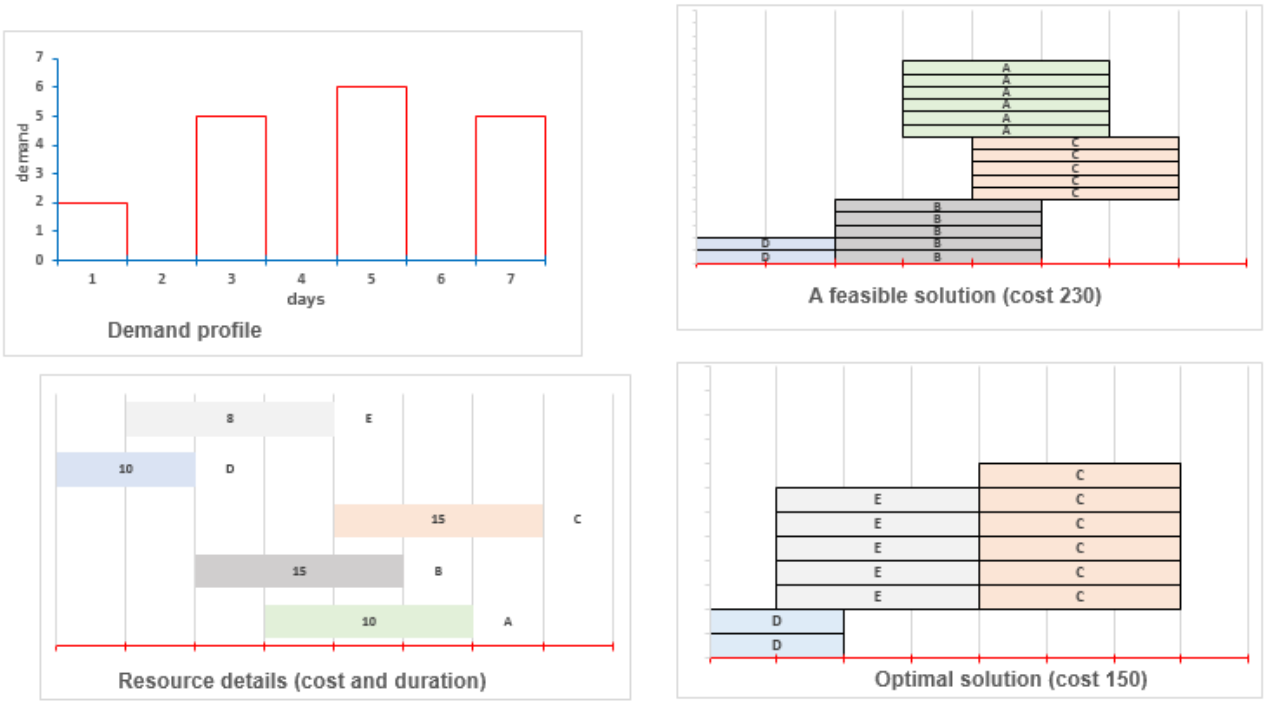

FIGURE 1. Illustration of S_RA problem.

Given A set $I = \{i\}$ of resources. Resource i has start time s_i, finish time f_i, cost c_i, and capacity h_i. It is said to be active over the interval $[s_i, f_i]$. Multiple copies of each resource is available to the decision maker. A time horizon partitioned into intervals 1 through T. Interval t has a resource demand of d_t (without loss of generality, d_t is assumed to be integer)

Required To find a least cost use of resources to satisfy all the resource requirements over the time period such that for each time slot only one resource contributes to the demand of the slot.

Note that for the S_RA problem, there is an additional constraint in the primal formulation that the demand for a time interval is met by just one of the resources. A feasible solution to the S_RA problem is in the form of a multi-set of S of resources. It finds application in several contexts such as CPU scheduling, routing, manufacturing and cloud computing.

Consider a scenario for a power generating firm where there are 5 options to produce energy. Resource A can be used to produce electricity at a rate of \$10 per unit for the 4th to the 6th day. Resources B and C are available from the 3rd to the 5th day and from the 5th to the 7th day respectively; both cost \$15 per unit. Resource E is active from the 2nd to the 4th day and costs \$8 per unit. Resource D is the option for shorter duration available only for the first 2 days. It costs \$10 per unit. The resources have fixed durations. Figure 1 shows the demand profile over 7 days. On the right side of Figure 1, two feasible solutions are shown.

The S_RA problem applies to many other real-life scenarios. For example, a cloud computing firm may announce different options of computing slots of various durations and costs. The problem faced by an individual user would be to meet his/her workload at minimum cost. Here the cloud computing firm may restrict individual users to combine different options (offers) for each slot to improve its revenue. Hence the problem would have SRA constraints. The framework is quite general and captures scenarios arising in domains such as workforce management, cloud computing, energy management and distributed computing.

The RA problem is a general version of the S_RA problem where there is no SRA constraint. The version of RA where a resource can be used at most once, it is called ZRA. We assume that no resource is active only over a strict subset of another resource's active period.

Sometimes, it may not be possible of the power generating firm to produce a feasible solution given the resources. In such cases, a restricted variant considers the partial covering scenario, wherein the input also specifies a number k and a feasible solution is only required to satisfy the demand for at least k time slots. In workforce management setting, this corresponds to the concept of meeting service level agreements (SLAs) for a significant fraction of times.

1.2. Related work

A variant of the RA problem uses packing constraints and is called the Unsplittable Flow (UF) problem. This problem has been widely studied in the literature. The special case of uniform bandwidth has been considered in [5]. where a constant factor approximation algorithm has been proposed. However, the general case of variable bandwidth is considerably more difficult. Bansal *et al.* [4] presented a quasi-PTAS for the UF problem under the restriction that all the capacities and demands are quasi-polynomial. Bonsma *et al.* [7] designed a polynomial time $(7+\epsilon)$-approximation algorithm with a "no-bottleneck assumption". Chekuri *et al.* [14] have proposed a $(2+\epsilon)$-approximation algorithm. A $(2+\epsilon)$-approximation algorithm has been proposed in [3] for the UFP on paths. Further, another constant factor approximation algorithm proposed in [1] works even without any restriction on the capacity of the resources. The knapsack covering problem, which is a relaxation of the RA problem, is known to be NP-hard but not in the strong sense. It admits FPTAS ([16,19]). There have been recent efforts to consider alternate linear programming(LP)-based algorithms for the unbounded minimum knapsack problem [8,9]. A primal dual based approach for the RA problem has been proposed by [12].

Solving the ZRA problem requires more complicated flow cover inequalities in order to derive constant factor approximation. Chakrabarty *et al.* [10] have studied a more general version of the ZRA problem called column restricted covering integer programs. Their framework yields a 40-approximation algorithm for the ZRA problem. Korula (see [10]) as well as Chakravarthy *et al.* [13] have presented a 4-approximation algorithm for the ZRA. Results of [5] imply a 4-approximation for column-restricted line cover problem. To the best of our knowledge, resource allocation with SRA constraints has been considered in [13] only.

The S_RA problem is a generalization of the covering version of the knapsack problem in which items may be selected multiple times. and is thus NP-hard [18]. Chakravarthy *et al.* [13] obtained a primal-dual based 16-approximation algorithm for the S_RA problem. Our technique is a generalization of the primal-dual algorithm of [13]. First, unlike [13] where time slots are considered in arbitrary order in deletion phase, we adjust the dual variables in the reverse order of the first phase. Second, during the deletion phase, we not only subtract from the primal variables, but also change the dual solution. In certain cases a dual variable is reduced to zero by increasing the two neighboring variables by the same amount. This approach helps in improving approximation ratio for any primal-dual based approximation approach, and has its own merit. The two generalizations make it possible to improve the approximation ratio from 16 to 12. The primal-dual paradigm was originally developed by Bar-Yehuda and Even [6] and Chvátal [15] for the weighted vertex cover and set cover problems. Of late, the primal-dual algorithms have been widely used in designing approximation algorithms ([2,17]).

1.3. Our results

The S_RA problem is NP-hard. So far, only 16-approximation to this problem is known. In this paper we propose a new combinatorial algorithm that improves the approximation ratio to 12. The algorithm is based on primal dual paradigm where the dual variables are adjusted through an innovative scheme. Using this, we could improve the approximation ratio of the partial covering version of the S_RA from 16 to 12.

2. Formulation of he S_RA problem

In this section, we present the LP formulation for the problem and its dual. This will pave the way for devising the primal-dual algorithm to be discussed subsequently.

Let $A(t)$ denotes the set of resources that are active at time interval t. The decision variable x_i in the primal denotes the number of copies of resource i used in the solution, and $h_{it} = min\{h_i, d_t\}$. An ILP formulation of the RA problem is as given below.

$$\text{Primal: Minimize } \sum_{i\in\mathcal{I}} c_i x_i \text{ subject to } \sum_{i\in A(t)} h_{it}x_i \geq d_t \ , \forall t \quad \text{and} \quad x_i \in \mathcal{Z}, \forall i \in 1\dots T. \tag{2.1}$$

$$\text{Dual: Maximize } \sum_{1\leq t\leq T} y_t d_t \text{ subject to } \sum_{t:i\in A(t)} h_{it}y_t \leq c_i, \forall i \quad \text{and} \quad \forall y_t \geq 0, \forall t \in 1\dots T. \tag{2.2}$$

Note that for the S_RA problem, there is an additional constraint in the primal formulation that the demand for a time interval is met by just one of the resources. A feasible solution to the S_RA problem is in the form of a multi-set S of resources. Let $f_S(i)$ denote the number of times i is used in S. The multi-set S can cover a time slot t, if there exists a resource $i \in A(t)$ such that $f_S(i) \times h_{it} \geq d_t$. The multi-set S is said to be a full cover, if S covers all the time slots. The cost of the multi-set S is given by $cost(S) = \sum_i f_S(i) \times c_i$. For a quick reference, notations used in this paper have been summarized below.

Table 1. Problem classes and notations.

RA	(Resource Allocation Problem)	Problem of allocating single or multiple copies of resources from a set of given resources to complete a given set of jobs over a time horizon at minimum cost
SRA	(Single Resource Assignment)	Constraint that in each time slot only one resource can be used (in multitude if required)
S_RA	(RA Problem with SRA constraint)	Resource Allocation Problem with additional SRA constraint
ZRA	(Zero One RA Problem)	Resource Allocation Problem with the constraint that a resource can be used at most once

t	$\triangleq$	index for time slot or time interval $(1\dots T)$	$x_i, f_A(i)$	$\triangleq$	number of copies of resource i used in the solution
i	$\triangleq$	index for resources (from set I)	$\widetilde{H}(i,t)$	$\triangleq$	adjusted bundle height at t met by the resource i $(= x_i h(i,t))$
$A(t)$	$\triangleq$	set of resources active at time interval t	A	$\triangleq$	primal solution after construction phase
d_t	$\triangleq$	demand at time slot t	S'	$\triangleq$	set resource in use after first for loop of reverse delete phase
s_i	$\triangleq$	start time of resource i			
f_i	$\triangleq$	finish time of resource i	B_j	$\triangleq$	set of resources selected for band j in a main time slot
c_i	$\triangleq$	per unit cost of resource i			
h_i	$\triangleq$	capacity of resource i	B	$\triangleq$	final primal solution
h_{it}	$\triangleq$	$min\{h_i, d_t\}$			

2.1. An approximation Algorithm for the S_RA problem

We reproduce working of the primal dual algorithm in [13] to make this article self-contained. The algorithm operates in three phases. In the construction phase, a feasible SRA solution is constructed. The algorithm considers the time slots in a greedy order (basically a time slot t with the highest demand is selected first). The dual variable y_t is increased to the maximum possible value until some dual constraint, say corresponding

to an interval i, becomes tight. $\lceil \frac{d_t}{h_{it}} \rceil$ copies of i are added to the solution. All the time slots in the span of the interval are marked as *dead*. In this case, t is a main time slot and i is the resource associated with t. These steps are iterated till demands for all the slots are met.

Step 0. Initialization Set $D_t = d_t \ \forall t$. Set $y_t = 0 \ \forall t$.
Step 1. Forward For each time slot $t \in \{1, \ldots, T\}$ considered in non-increasing order of demands d_t,

(1) **If** t is not dead,
   (a) Increase y_t to the maximum possible values while maintaining dual feasibility. Let i be the resource for which the dual constraint becomes tight first.
   (b) Set $x_i \leftarrow \lceil \frac{d_t}{h_{it}} \rceil$.
   (c) Mark time slot t as "main slot associated with i". Mark all other slots in the span of i as "dead". (Main time slot t and all other slots marked dead by i are associated with i.)
   end if

end for

Let the SRA solution returned by the construction phase is A. For an interval i and a timeslot t, $\widetilde{H}(i,t) = x_i h(i,t)$ denotes the adjusted bundle height at t met by the resource i(note $x_i = f_A(i)$). It is easy to see that $\widetilde{H}$ satisfies the following property (due to knapsack cover issue).

Lemma 2.1 [13]. *Let i be any interval chosen in A and t be any main time slot within the span of i. Then, at the end of the construction phase, $\widetilde{H}(i,t^*) \leq 2d_t$.*

In the deletion phase, certain redundant intervals are deleted from A to obtain a multi-set S'. The main time slots are considered in an arbitrary order. Let t be a main time slot. The set of intervals active at $t^*(A(t^*))$ are divided into a set of bands in a geometric fashion as follows. Let $m = \lceil \log 2d_t \rceil$. For $1 \leq j \leq m$, define the band B_j to be

$$B_j = \left\{ i \in A \cap A(t) : \frac{2d_t}{2^j} < \widetilde{H}(i,t) \leq \frac{2d_t}{2^{j-1}} \right\}.$$

For each band B_j, let i_1 be the interval in B_j extending farthest to the left (*i.e.*, having the minimum start-time); similarly, let i_2 be the interval in B_j extending farthest to the right (*i.e.*, having the maximum end-time). Only these two intervals are retained and all the other interval (bundles) of B_j are deleted from A. Let the constructed multi-set be S'. Though S' need not be a full cover, it has been established that S' meets at least half of the demand in SRA manner. Due to the fact that (i) the leftmost and rightmost interval within each band are retained and (ii) geometric series of the bands, the following holds true.

Lemma 2.2 [13]. *At the end of the deletion phase, for any time slot t, there exists an interval $i' \in S'$ such that i' is active at t and $f_{S'}(i')h_{i'} \geq \frac{d_t}{2}$.*

In the last phase called the doubling phase, solution S' is converted into an SRA full cover B, by simply doubling the number of copies of every resource in S'. A 16-approximation (2 factor each from geometric series, knapsack cover, retaining both leftmost and rightmost intervals in each band and doubling) is obtained.

Lemma 2.3 [13]. *For any main time slot t, $\sum_{i \in A(t^*)} h_{it^*} f_B(i) \leq 16 d_{t^*}$*

3. An improved approximation Algorithm for the S_RA problem

We will use the construction phase of [13] to generate a feasible primal solution for the S_RA problem. However, we modify the deletion phase. We refer to this second phase as Reverse Delete phase.

During the construction phase for the example in Figure 1, the following are the values to dual variables corresponding to the main time slots: $y_5 = 10, y_3 = 5, y_7 = 5, y_1 = 10$. The associated multi-set of resources used are: {6A, 5B, 5C, 2D}. The constructed solution is as shown in the upper right section of Figure 1.

Before we present the modified reverse delete phase, we need a few routines. Routine left-estimate estimates the maximum possible decrease of y_t for a time slot t without violating dual feasibility for the resources to the left of t. Besides the maximum allowable change of y_t, it also returns r_l (the constraining resource which is blocker for further decrease of y_t), T (a chain of main-time slots towards left of t) and J (the associated tight resources).

Routine: Left-estimate(time slot t, associated resource i, adjustment quantity δ)

(1) $T = \phi$, $J = \phi$. Assume y_t to be 0. $f = True$. $cur_t = t$
(2) **While** there is a main timeslot (say, t_b with the associated resource j_a) to the left of cur_t and j_b is active at cur_t and dual constraint for j_b is not tight (select rightmost t_b in case there are multiple candidates)
  (a) Pre-append t_b to T, j_b to J.
  (b) **If** $f = True$
    (i) Assume $y_{t_b} = y_{t_b} + y_t$
    (ii) **If** Dual feasibility is violated for any resource active over t_b
      (A) Let δ be the minimum amount of violation. Let r_l be the rightmost minimum violating resource.
      (B) Return $[y_t - \delta, r_l, T, J]$
  (c) **Else**
    (i) Assume $y_{t_b} = y_{t_b} - y_t$
  (d) $cur_t = t_b$
  (e) $f = \bar{f}$
(3) **end while**

Routine right-estimate estimates maximum possible decrease of y_t for a time slot t without violating dual feasibility for the resources to the right of t. Besides the maximum allowable change of y_t, it also returns the constraining resource r_r which is blocker for further decrease of y_t, T (a chain of main-time slots towards right of t) and J (the associated tight resources).

Routine: Right-estimate(time slot t, associated resource i)

(1) $T = \phi$, $J = \phi$. Assume y_t to be 0. $f = True$. $cur_t = t$
(2) **While** there is a main timeslot (say t_a with the associated resource j_a) to the right of cur_t and j_b is active at cur_t and dual constraint for j_a is not tight (select leftmost t_a in case there are multiple candidates)
  (a) Post-append t_a to T, j_a to J.
  (b) **If** $f = True$
    (i) Assume $y_{t_a} = y_{t_a} + y_t$
    (ii) **If** Dual feasibility is violated for any resource active over t_a
      (A) Let δ be the minimum amount of violation. Let r_r be the leftmost minimum violating resource.
      (B) Return $[y_t - \delta, r_r, T, J]$
  (c) **Else**
    (i) Assume $y_{t_a} = y_{t_a} - y_t$
  (d) $cur_t = t_a$
  (e) $f = \bar{f}$
(3) **end while**

Routine adjust-dual-left is used to adjust the dual variables without violating dual feasibility of any resources to the left of t. It uses the maximum possible decrease in the y_t value as well as the chains of main-time slots and the associated tight resources (J) as input.

Routine: Adjust-dual-left(time slot t, associated resource i,δ, T_L, J_L)

(1) $f = True$.
(2) **While** T_L is not empty
  (a) **If** $f = True$
    (i) Remove the last element of T_L. Let this element be cur_t.
    (ii) Increment y_{cur_t} by δ
  (b) **Else**
    (i) Decrement y_{cur_t} by δ
  (c) $f = \bar{f}$
(3) **end while**

Routine adjust-dual-right is used to adjust the dual variables without violating dual feasibility of any resources to the right of t. It uses the maximum possible decrease in the y_t value as well as the chains of main-time slots and the associated tight resources (J) as input.

Routine: Adjust-dual-right(time slot t, associated resource i, δ, T_L, J_L)

(1) $f = True$.
(2) **While** T_L is not empty
  (a) **If** $f = True$
    (i) Remove the first element of T_L. Let this element be cur_t.
    (ii) Increment y_{cur_t} by δ
  (b) **Else**
    (i) Decrement y_{cur_t} by δ
  (c) $f = \bar{f}$
(3) **end while**

Routines adjust-primal-left and adjust-primal-right are two important routines. These routines are designed to maintain primal complementary conditions and primal feasibility of the solution. Again left and right variants are used to adjust the values of primal and dual values for the jobs to the left of t and to the right of t respectively. In these routines, $first(L)$ and $last(L)$ indicate the first and the last element respectively of a list L. $Next(j, L)$ and $prev(j, L)$ are used to obtain the next and the previous element of j respectively from L.

Routine: Adjust-primal-left(time slot t, associated resource i, δ, r_l, T_L, J_L)

(1) **While** T_L is not empty
  (a) Let $cur_j = first(J_L)$. $j_l = r_l$, $j_r = next(cur_j, J_L)$, $cur_t = first(T_L)$, $t_r = next(cur_j, T_L)$.
  (b) **If** cur_j is not required to meet demand of any dead slot where j_l and j_r are not active
    (i) Set $x_{cur_j} = 0$. Mark cur_j as inactive. Remove cur_j from S'.
    (ii) Set x_{j_l} to meet the demand for the slot cur_t. Mark j_l as the active resource associated with the slot cur_t.
    (iii) Adjust x_{j_r} to meet the demand for the slot t_r. Add j_r to S'.
  (c) **else If** cur_j is required to meet the demand of any dead slot t_u (where both j_l and j_r are not active)
    (i) Mark t_u as the main slot with the associated resource cur_j. Adjust x_{cur_j} to meet the demand for the slot t_u. Add cur_j to S'.
    (ii) Set x_{j_l} to meet the remnant demand for the slot cur_t. Mark j_l as the active resource associated with the slot cur_t. Add j_l to S'.
    (iii) **If** there is remnant non-zero demand for the slot t_r

(A) Adjust x_{j_r} to meet the remnant demand for the slot t_r. Add j_r to S'.
(iv) **else If** $t_r = t$
(A) Set $y_{t_u} = y_t$, set y_t to zero
(v) **end if**
(d) **end if**
(e) Remove the first two elements from each of T_L and J_L.
(2) **end while**

Routine: Adjust-primal-right(tme slot t, associated resource i, δ, r_r, T_R, J_R)

(1) **While** T_R is not empty
(a) Let $cur_j = last(J_R)$. $j_r = r_r$, $j_l = prev(cur_j, J_R)$, $cur_t = last(T_R)$, $t_l = prev(cur_j, T_R)$.
(b) **If** cur_j is not required to meet demand of any dead slot where j_l and j_r are not active
(i) Set $x_{cur_j} = 0$. Mark cur_j as inactive. Remove cur_j from S'.
(ii) Set x_{j_r} to meet the demand for the slot cur_t. Mark j_r as the active resource associated with the slot cur_t.
(iii) Adjust x_{j_l} to meet the demand for the slot t_l. Add j_l to S'.
(c) **else If** cur_j is required to meet the demand of any dead slot t_u (where both j_l and j_r are not active)
(i) Mark t_u as the main slot with the associated resource cur_j. Adjust x_{cur_j} to meet the demand for the slot t_u. Add cur_j to S'.
(ii) Set x_{j_r} to meet the remnant demand for the slot cur_t. Mark j_r as the active resource associated with the slot cur_t. Add j_r to S'.
(iii) **If** there is remnant non-zero demand for the slot t_l
(A) Adjust x_{j_l} to meet the remnant demand for the slot t_l. Add j_l to S'.
(iv) **else If** $t_l = t$
(A) Set $y_{t_u} = y_t$, set y_t to zero.
(v) **end if**
(d) **end if**
(e) Remove the last two elements from each of T_R and J_R.
(2) **end while**

We are now ready to present the modified reverse delete phase.

(1) [**Step 2: Reverse Delete**]
(2) $S' = \phi$.
(3) **For** each main time slot $t \in \{1, \ldots, T\}$ considered in the reverse of the order they were considered in the forward phase,
(a) Let i be the resource associated with t. Let $B_j = \{i \in A \cap A(t) : \frac{2d_t}{2^j} < \widetilde{H}(i,t) \leq \frac{2d_t}{2^{j-1}}\}, 1 \leq j \leq m = \lceil \log 2d_t \rceil$.
(b) **For** $j = 1, \ldots, m$ do
(i) Let i_1 be the interval in B_j extending farthest to the left (*i.e.*, having the minimum start-time); similarly, let i_2 be the interval in B_j extending farthest to the right (*i.e.*, having the maximum end-time). Only these two intervals from B_j are added to S'. Set $x_r = 0$ for all other $r \in B_j$.
(c) **end for**
(4) **end for**
(5) **For** each main time slot $t \in \{1, \ldots, T\}$ considered in the reverse of the order they were considered in the forward phase,
(a) Let i be the resource associated with t.
(b) **If** $i \notin S'$

(i) Let t_l and t_r be the main time slots (different from t) closest to t on the left and the right respectively such that the associated resources j_l and j_r are active at t and $x_{j_l}, x_{j_r} > 0$
(ii) $[\delta_L, r_l, T_L, J_L]$ =Left-estimate(t)
(iii) $[\delta_R, r_r, T_R, J_R]$ =Right-estimate(t)
(iv) $y_m = \min\{\delta_R, \delta_L\}$
(v) Adjust-dual-left(y_m, T_L, J_L)
(vi) Adjust-dual-right(y_m, T_R, J_R)
(vii) $y_t = y_t - y_m$.
(viii) **If** $y_m > 0$
(A) mark t as `borrowed and main`.
(B) Adjust-primal-left(y_m, T_L, J_L)
(C) Adjust-primal-right(y_m, T_R, J_R)
(ix) **end if**
(c) **end if**
(6) **end for**

With reference to the running example, $B_1 = \{D\}, \{B\}, \{B, C\}$ and $\{C\}$ for the main time slots $1, 3, 5$ and 7 respectively after the first for loop in the reverse delete. Hence $S' = \{D, B, C\}$. Note that S' does not contain the resource A (as when considering main slot 5, A appears in the same band as B and C and B is the resource extending farthest to the left while C is the resource extending farthest to the right). After the reverse delete step, the dual variables will be $y_5 = 7, y_3 = 8, y_7 = 8, y_1 = 10$. Time slot 5 is marked borrowed and main. Final resources used will be a multi-set of D, B and C.

Based on the working of the algorithm, we can establish following properties easily.

Lemma 3.1. *At the end of the reverse delete phase, for any time slot t, there exists an interval $i' \in S'$ such that i' is active at t and $f_{S'}(i')h_{i'} \geq \frac{d_t}{2}$.*

Proof. We first prove that the lemma holds at the end of line 10 of the reverse delete. Let i be the interval associated with t. If $i \in S'$, then $f_{S'}(i)h_i \geq \frac{d_t}{2}$. Hence, we can take $i' = i$ and the lemma holds true. Let us assume that i got deleted while considering a main time slot t^*. Now consider the demand fulfillment at t^*. Since i got deleted, the algorithm must have retained a leftmost and a rightmost interval (say i_l and i_r respectively) that were in the same band as i. At least one of these two intervals (say i_l) must be active at t. Hence, $H(i_l, t^*) \geq H(i, t^*)/2$. There are two possible cases: (i) $h_i < d_{t^*}$ and (ii) $h_i \geq d_t$. For case (i), $H(i, t^*) = f_{S'}(i)h_i \geq d_t$ as i is associated with t. Hence, from the last two expressions, $H(i_l, t^*) \geq \frac{d_t}{2}$. So with $i' = i_l$ the lemma holds. For case (ii), $H(i, t^*) \geq d_{t^*}$, and hence i belongs to the band $[d_{t^*}, 2d_{t^*}]$. In other words, i belongs the the same band as i_l and $H(i_l, t^*) \geq d_{t^*}$. Let t' be the main time slot for i. As both t^* and t' are main time slots, $d_{t^*} \geq d_{t'}$ otherwise in the construction phase t' would not have been considered earlier and it would have marked t^* dead. As t' is the main slot for i, $d_{t'} \geq d_t$. It follows that $d_{t^*} \geq d_t$. Therefore, $f_{S'}(i_l)h_{i_l} \geq d_t$. Again for $i = i_l$, the lemma is true.

Now we would like to point out that the rest of the reverse delete does not violate the lemma. In other words, execution of routines adjust-primal-left and adjust-primal-right does not violate the lemma. During the execution of the lemma, whenever a resource from S' is deleted, one or more resources are added to make sure that the contribution for deleted resource is adequately met for each of the effected time slot. Hence at the end of the reverse delete the lemma holds true. □

We use the doubling phase of [13] without any modification. The final set of resources used is referred to as B and the dual solution is y.

Lemma 3.2. *At the end of the primal dual algorithm, the solution (B, y) satisfies* (1) *dual feasibility,* (2) *primal feasibility,* (3) *primal complementary slackness,* (4) *the approximate dual complementary condition for any main slot t^*, $y_t > 0 \implies \sum_{i \in A(t^*)} h_{it^*}x_i \leq 10d_{t^*}$, and* (5) *the approximate dual complementary condition*

for a **`borrowed and main`** *time slot t with the two closest* **`self`** *slots l^t and r^t: $y_t > 0 \implies \sum_{i \in A(t)} w_{it} x_i \leq 8(d_{l^t} + d_{r^t}) \leq 16 d_t$. Further* (6) *for a dead timeslot t, $y_t = 0$ and* (7) *For each borrowed and main slot t, there are two closest main slots l^t and r^t such that $y_t \leq \min\{y_{l^t}, y_{r^t}\}$ or $y_t = 0$.*

Proof.

(1) Let us consider effect of change of y_t for a main slot on dual feasibility of the resources on or to the left of t. Routine adjust-dual-left considers all the tight resources on the left of t whose dual feasibility may be effected by change in y_t and maintains them tight. Amount of decrement of the dual value y_s for any slot s is upper bounded by the $y_m = \min_{h \in T : h}$ is at even position in $_T\, y_h \leq y_s$. Hence, if y_s is decremented, post adjustment $y_s \geq 0$.
Amount of increment of a dual value y_s cannot be arbitrary. As for each increment, we are checking the dual feasibility of all the resources active over s, dual feasibility cannot be violated due to increase of y values.
For any tight resource r, $|T \cap A(r)|$ cannot be odd. Assume this to be odd. Let us consider the leftmost slot of r that is added to T. Then for this slot when t_r is considered, the while condition of the left-estimate routine becomes false (as resource r is already tight) and it will not be added to T. A contradiction. If $|T \cap A(r)|$ is even, $\sum_{i \in A(r)} \Delta y_i = 0$, since each $|\Delta y_h| = y_m, \forall h \in T$ and adjustment of y alternates in sign. So the dual constraint for r remains tight post execution of the routine.
A similar argument holds true for change of y_t for a `self` slot on dual feasibility of the resources on or to the right of t (through routine.

(2) Primal variable x_i is increased to satisfy the primal constraints for a time slot in SRA manner. We continue the process till we satisfy primal constraints for all the time slots. Since the time slots are considered in non-increasing order of demand, if there is any time slot with unmet demand at an intermediate step and there is a feasible solution, we will always be able to increase y values corresponding to this time slot without violating dual feasible constraints. During the reverse delete phase, we can violate the primal feasibility up to a factor of $\frac{1}{2}$. Hence, at the end of the doubling phase (end of algorithm), B is primal feasible.

(3) Primal complementary slackness condition is true at the end of the forward phase as per [12]. If the value of a primal variable becomes zero during the reverse delete phase, then the condition is vacuously true. Also, tight resources remain tight when we adjust the y-values during the reverse delete phase. In routines adjust-primal-left and adjust-primal-right, a primal variable is set to non-zero value only when the corresponding dual constraint is tight.

(4) For a main time slot t^*, let the associated interval be i. In this case, $i \in B$. Then $\sum_{i' \in S' \cap A(t^*) \setminus \{i\}} h_{i't^*} x_{i'} \leq \sum_{j=2}^{m} 2 \times (\frac{2d_{t^*}}{2^{j-1}}) \leq 4 d_{t^*}$, as $m = \lceil \log(2d_{t^*}) \rceil$. Therefore, $\sum_{j \in B \cap A(t^*)} h_{jt^*} x_j \leq 2d_{t^*} + 8d_{t^*}$, (as the set of retained resources from each band can span t from left or right) $= 10 d_{t^*}$.

(5) For this case, $x_i > 0$ and $i \notin B$. In this case, total contribution of the jobs to the time slot $t \leq \sum_{j=1}^{m_l} 2 \times (\frac{2d_{t_l}}{2^{j-1}}) + \sum_{j=1}^{m_r} 2 \times (\frac{2d_{t_r}}{2^{j-1}})$, as $m_l = \lceil \log(2d_{t_l}) \rceil$ and as $m_r = \lceil \log(2d_{t_r}) \rceil$
$\leq 8(d_{t_l} + d_{t_r}) \leq 16 d_{t^*}$.

(6) For a dead timeslot, we never modify the dual variable y_t. Hence $y_t = 0$.

(7) By the working of the reverse delete phase, a time slot is marked borrowed and main only when the said condition is true. □

Theorem 3.3. *The proposed algorithm is a polynomial time approximation scheme that finds an SRA full cover B such that $cost(B) \leq 12 \times OPT$.*

Proof.
Here, $OPT \leq \sum_{i \in \mathcal{I}} c_i x_i = \sum_{i \in \mathcal{I} : x_i > 0} x_i (\sum_{1 \leq t \leq T} y_t h_{it})$, by primal complementarity condition $\leq \sum_{1 \leq t \leq T : y_t > 0} y_t (\sum_{i \in A(t) :: x_i > 0} h_{it} x_i)$, by rearranging the terms $\leq \sum_{1 \leq t \leq T : t \text{ is main}} y_t (\sum_{i \in A(t) : x_i > 0} h_{it} x_i) + \sum_{1 \leq t \leq T : t \text{ is borrowed and main}} y_t (\sum_{i \in A(t) : x_i > 0} h_{it} x_i) = \sum_{1 \leq t \leq T : t \text{ is main}} 10 y_t d_t + \sum_{1 \leq t \leq T : t \text{ is borrowed and main}} 8 y_t (d_{t_l} + d_{t_r})$, where l^t and r^t are the

two closest `self` slots $\leq \sum_{1\leq t\leq T:t \text{ is main}} 10y_t d_t + \sum_{1\leq t\leq T:t \text{ is } \texttt{borrowed and main}} 6y_t(d_{l^t} + d_{r^t}) + \sum_{1\leq t\leq T:t \text{ is } \texttt{borrowed and main}} 2y_t(d_{l^t} + d_{r^t}) \leq \sum_{1\leq t\leq T:t \text{ is } \texttt{main}} 12y_t d_t + \sum_{1\leq t\leq T:t \text{ is } \texttt{borrowed and main}} 6y_t(d_{l^t} + d_{r^t})$, as $y_t \leq \min\{y_{l^t}, y_{r^t}\} \leq \sum_{1\leq t\leq T} 12y_t d_t$, as $d_{l^t} \leq d_t$ and $d_{r^t} \leq d_t = 12OPT$.

Since there are bounded number of time slots with non-zero demands and in each cycle of the construction phase demand for at least one time slot is met, there is only a finite number of steps. The reverse delete stage also invokes a polynomial number of iterations. Hence, the algorithm has polynomial time complexity. □

4. Further results

An SRA k-partial cover S is a k-partial cover if S covers at least k time slots in an SRA fashion. Let OPT and $pOPT$ be the optimal value of full cover and k-partial cover, respectively.

Corollary 4.1. *There exists an SRA k-partial cover B such that $cost(B) \leq 12 \times pOpt$.*

Proof. The proof of the corollary is similar that outlined in [13]. Given a subset of time slots $X \subseteq [1, T]$, the primal-dual algorithm can produce a solution S such that S covers all the time slots in X in an SRA fashion and $cost(S) \leq 12cost(Opt(X))$, where $Opt(X)$ is the optimum solution covering all the time slots in X. Let the optimum partial k-cover covers a subset of time slots Z with $|Z| \geq k$. We consider a modified instance of the problem where the demands of all the slots other than Z are set to zero. By applying the proposed algorithm, we can obtain a solution S' such that S' covers all the time slots in Z in an SRA fashion and $cost(S') \leq 12pOpt$. Hence the corollary follows with $B = S'$. □

In [13], the set $A(t)$ of intervals active at t were partitioned in a geometric fashion into a set of bands starting from highest value of $2d_t$ resulting in $\lceil \log(2d_{t^*}) \rceil$ bands. Band $B_j = \{i \in S \cap A(t) : \frac{d_{t^*}}{2^j} < H(i, t^*) \leq \frac{d_{t^*}}{2^{j-1}}\}$. However, we consider the bands starting with lowest non-zero demand in the slots in the span of $A(t)$, say $d^t_{\min}$, resulting in $\lceil \log(\frac{2d_{t^*}}{d^t_{\min}}) \rceil$ bands: band $B_j = \{i \in S \cap A(t) : 2^j d^t_{\min} \leq H(i, t^*) < 2^{j-1} d^t_{\min}\}$. This immediately improves the performance, with improvement depending on ratio of the largest to smallest non-zero demands in overlapping region (often referred to as mountain). This observation leads to better ratio for the proposed algorithm as well as for algorithm in [13].

5. Conclusion

We are not sure whether the current approximation ratio of 12 is tight for the S_RA problem. Hence, it would be worthwhile to prove approximation hardness for the S_RA problem. We feel that the approximation ratio can be improved further before hitting approximation hardness. Taking a cue from this, it may be possible to prove a better approximation ratio for the RA problem.

Acknowledgements. The author would like to thank Diptesh Ghosh and anonymous reviewers for their constructive feedback.

References

[1] A. Adamaszek and A. Wiese, A quasi-PTAS for the Two-Dimensional Geometric Knapsack Problem, in *Proc. Symp. Discrete Algorithms* (2015) 1491–1505.

[2] A. Agrawal, P.N. Klein and R. Ravi, When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput.* **24** (1995) 440–456.

[3] A. Anagnostopoulos, F. Grandoni, S. Leonardi and A. Wiese, A Mazing ϵ Approximation for Unsplittable Flow on a Path, in *Proc. Symp. Discrete Algorithms* (2014) 26–41.

[4] N. Bansal, A. Chakrabarti, A. Epstein and B. Schieber, A quasi-PTAS for unsplittable flow on line graphs, in *Proc. Symp. Theory Comput.* (2006) 721–729.

[5] A. Bar−Noy, R. Bar-Yehuda, A. Freund, J. Naor and B. Schieber, A unified approach to approximating resource allocation and scheduling. *J. ACM* **48** (2001) 1069–1090.

[6] R. Bar−Yehuda and S. Even, A linear time approximation algorithm for the weighted vertex cover problem. *J. Algorithms* **2** (1981) 198–203.

[7] P. Bonsma, J. Schulz and A. Wiese, A constant factor approximation algorithm for unsplittable flow on paths, in *Proc. Symp. Found. Comput. Sci.* (2011) 47–56.

[8] T. Carnes and D. Shmoys, Primal-dual schema for capacitated covering problems, in *Proc. Confer. Integer Program. Combinatorial Optimiz.* (2008) 288–302.

[9] R.D. Carr, L. Fleischer, V.J. Leung and C.A. Phillips, Strengthening integrality gaps for capacitated network design and covering problems, in *Proc. Symp. Discrete Algorithms* (2000) 106–115.

[10] D. Chakrabarty, E. Grant and J. Könemann. On column-restricted and priority covering integer programs, in *Proc. Confer. Integer Program. Combinatorial Optimiz.* (2010) 355–368.

[11] V. Chakaravarthy, S. Kenkre, S.A. Mondal, V. Pandit and Y. Sabharwal, Reusable Resource Scheduling *via* Colored Interval Covering, in *Proc. Inter. Confer. Parallel and Distributed Processing* (2016) 1003–1012.

[12] V. Chakaravarthy, A. Kumar, G. Parija, S. Roy and Y. Sabharwal, Minimum cost resource allocation for meeting job requirements, in *Proc. IEEE Inter. Parallel and Distributed Processing Symposium* (2011) 14–23.

[13] V. Chakaravarthy, A. Kumar, S. Roy and Y. Sabharwal, Resource Allocation for Covering Time Varying Demands, in *Proc. Europ. Symp. Algorithms* (2011) 543–554.

[14] C. Chekuri, M. Mydlarz and F. Shepherd, Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms* **3** (2007).

[15] V. Chvátal, A greedy heuristic for the setcovering problem. *Math. Oper. Res.* **4** (1979) 233–235.

[16] O. Ibarra and C. Kim, Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM* **22** (1975) 463–468.

[17] K. Jain and V. Vazirani, Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM* **48** (1975) 274–296.

[18] R. Karp, Reducibility among combinatorial problems. In Complexity of Computer Computations, edited by R. Miller and J. Thatcher. Plenum Press (1972) 85–103.

[19] E.L. Lawler, Fast approximation algorithms for knapsack problems. *Math. Oper. Res.* **4** (1979) 339–356.