Scheduling Resources for Executing a Partial Set of Jobs

Venkat Chakravarthy, Arindam Pal, Sambuddha Roy[1], Yogish Sabharwal

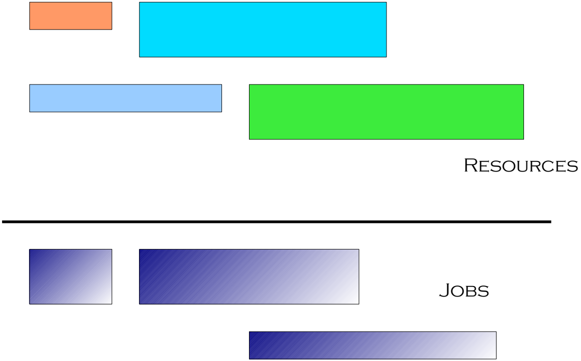
[1] sambuddha@in.ibm.com
IBM Research – India

FSTTCS 2012

A Typical Scheduling Scenario

- We are given a collection of jobs. Each job is an interval, with attributes: *starting time*, *ending time* and *capacity requirement*.
- We are also given a collection of resources; resources are also intervals. A resource interval has attributes *starting time*, *ending time*, *capacity* and also an associated *cost*.

An example scenario



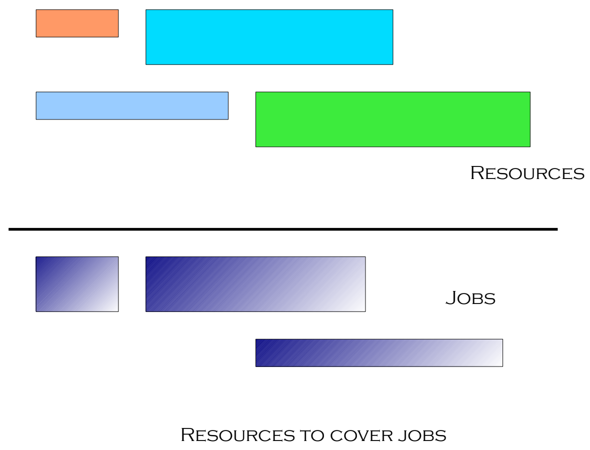
A Typical Scheduling Scenario

- When does a resource *cover* a job? If it
  - (a) spans the length of the job, and
  - (b) the capacity of the resource is sufficient to fulfill the capacity requirement of the job.
- Generalize this notion to *collection of resources*, and *collection of jobs*. A collection of resources *covers* a collection of jobs, if at any point of time, the total capacity of resources available is sufficient to fulfill the capacity requirements of the jobs active at that time point.
- **Overall Objective:** To fulfill (i.e. complete) the jobs using a *minimum cost* collection of resources.
- Call this the `ResAll` problem.

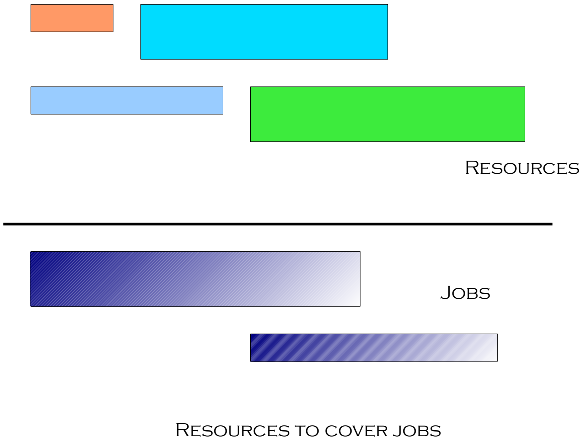
A Typical Scheduling Scenario

- When does a resource *cover* a job? If it
  - (a) spans the length of the job, and
  - (b) the capacity of the resource is sufficient to fulfill the capacity requirement of the job.
- Generalize this notion to *collection of resources*, and *collection of jobs*. A collection of resources *covers* a collection of jobs, if at any point of time, the total capacity of resources available is sufficient to fulfill the capacity requirements of the jobs active at that time point.
- **Overall Objective:** To fulfill (i.e. complete) the jobs using a *minimum cost* collection of resources.
- Call this the `ResAll` problem.

An example scenario



An example scenario



Two Variants of `ResAll`

- Resources may be picked at most once (the $\{0, 1\}$ version) or multiple times (the multiplicity version).
- Both variants have a 4-factor approximation [BarNoy et al.01].
- Typically however, the $\{0, 1\}$ version is at least as *hard* as the multiplicity version.

Two Variants of `ResAll`

- Resources may be picked at most once (the $\{0, 1\}$ version) or multiple times (the multiplicity version).
- Both variants have a 4-factor approximation [BarNoy et al.01].
- Typically however, the $\{0, 1\}$ version is at least as *hard* as the multiplicity version.

Two Variants of `ResAll`

- Resources may be picked at most once (the $\{0, 1\}$ version) or multiple times (the multiplicity version).
- Both variants have a 4-factor approximation [BarNoy et al.01].
- Typically however, the $\{0, 1\}$ version is at least as *hard* as the multiplicity version.

A Real Life Scenario

- Often it is the case that not all the jobs in the collection need to be fulfilled, but there are SLA requirements such as: “satisfy 90% of the jobs".
- Scenarios where such problems arise naturally: call center optimization, sensor networks, cloud computing, energy management . . .

A Real Life Scenario

- Often it is the case that not all the jobs in the collection need to be fulfilled, but there are SLA requirements such as: “satisfy 90% of the jobs".
- Scenarios where such problems arise naturally: call center optimization, sensor networks, cloud computing, energy management . . .

Problem Definition

- Formally, we are given a collection $\mathcal{R}$ of $m \in \mathbb{N}$ *resource intervals*. Each resource type has a cost per unit of the resource.
- Thus, a resource interval i is equipped with a "height" h_i (the *proficiency or capacity* of the resource) cost c_i, starting time s_i and ending time e_i.
- We are also given a collection $\mathcal{J}$ of $n \in \mathbb{N}$ *job intervals*. Each job interval j has a starting time s_j, ending time e_j and a height h_j (i.e. capacity requirement).

Problem Definition

- Formally, we are given a collection $\mathcal{R}$ of $m \in \mathbb{N}$ *resource intervals*. Each resource type has a cost per unit of the resource.
- Thus, a resource interval i is equipped with a “height" h_i (the *proficiency or capacity* of the resource) cost c_i, starting time s_i and ending time e_i.
- We are also given a collection $\mathcal{J}$ of $n \in \mathbb{N}$ *job intervals*. Each job interval j has a starting time s_j, ending time e_j and a height h_j (i.e. capacity requirement).

The SLA requirement

- We are also given a number k ($1 \leq k \leq n$), that corresponds to the SLA requirement: the number of *jobs* that need to be fulfilled.
- The **objective** is to select a minimum cost collection of resource intervals from $\mathcal{R}$ so that at least k of the jobs in $\mathcal{J}$ are satisfied.
- Call this problem the `PartialResAll` problem (it is NP-hard – Minimum Knapsack Cover is a special case).
- This is a *partial covering* problem, where only a certain fraction of the total collection of jobs need to be covered.

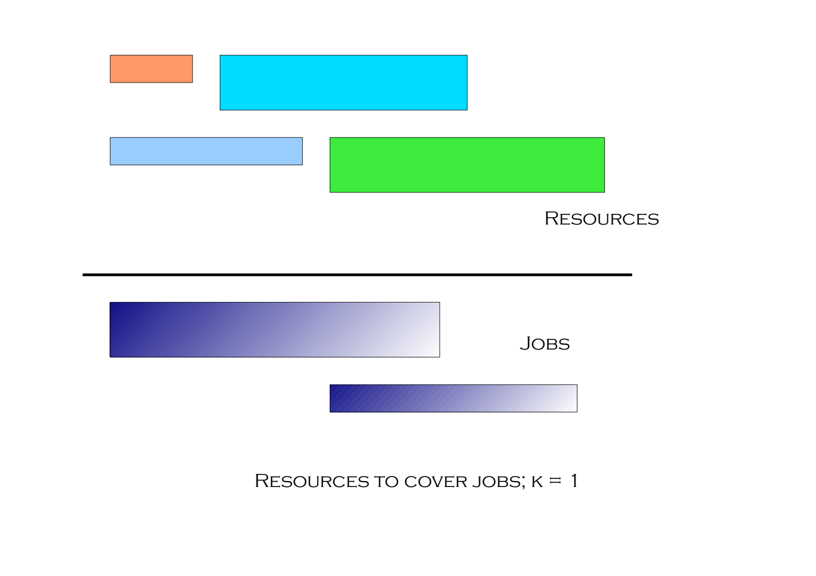
The SLA requirement

- We are also given a number k ($1 \leq k \leq n$), that corresponds to the SLA requirement: the number of *jobs* that need to be fulfilled.
- The **objective** is to select a minimum cost collection of resource intervals from $\mathcal{R}$ so that at least k of the jobs in $\mathcal{J}$ are satisfied.
- Call this problem the `PartialResAll` problem (it is NP-hard – Minimum Knapsack Cover is a special case).
- This is a *partial covering* problem, where only a certain fraction of the total collection of jobs need to be covered.

The SLA requirement

- We are also given a number k ($1 \leq k \leq n$), that corresponds to the SLA requirement: the number of *jobs* that need to be fulfilled.
- The **objective** is to select a minimum cost collection of resource intervals from $\mathcal{R}$ so that at least k of the jobs in $\mathcal{J}$ are satisfied.
- Call this problem the `PartialResAll` problem (it is NP-hard – Minimum Knapsack Cover is a special case).
- This is a *partial covering* problem, where only a certain fraction of the total collection of jobs need to be covered.

An example scenario



Ambient space of our problem

- Instances of covering problems abound: Set Cover, Vertex Cover, Minimum Spanning Tree, Steiner Network etc.
- Any such covering problem has a partial covering variant.
- The `PartialResAll` problem is the partial covering version of the `ResAll` problem.

Ambient space of our problem

- Instances of covering problems abound: Set Cover, Vertex Cover, Minimum Spanning Tree, Steiner Network etc.
- Any such covering problem has a partial covering variant.
- The `PartialResAll` problem is the partial covering version of the `ResAll` problem.

History of Partial Covering problems

- 2-factor approximations for the Partial Vertex Cover problem [Bar-Yehuda99, Gandhi-Khuller-Srinivasan01, Mestre07].
- The k-Median problem has constant factor approximations; methods are based on Lagrangian Relaxations [cf. JainVazirani99]
- 2-factor approximations for the k-MST problem [Garg05]
- etc. . .

Holy Grail

- The most general problem in this direction is to prove a constant factor approximation for the $\{0, 1\}$ version of the `PartialResAll` problem.

Main Result

- Here we consider the following *restriction* of the `PartialResAll` problem:
  - The **multiplicity** version: resources are allowed to be picked multiple times.
  - Jobs have **unit** height.
- Call this problem the restricted `PartialResAll` problem.

Theorem

There is a polynomial time $O(\log \ell_{\max}/\ell_{\min})$-factor approximation for the restricted `PartialResAll` problem.

Here, $\ell_{\max}$ and $\ell_{\min}$ stands for the longest and shortest lengths of the **jobs**. Note that this can be easily seen to be bounded by n, the *number* of jobs.

Main Result

- Here we consider the following *restriction* of the `PartialResAll` problem:
  - The **multiplicity** version: resources are allowed to be picked multiple times.
  - Jobs have **unit** height.
- Call this problem the restricted `PartialResAll` problem.

Theorem

There is a polynomial time $O(\log \ell_{\max}/\ell_{\min})$-factor approximation for the restricted `PartialResAll` problem.

Here, $\ell_{\max}$ and $\ell_{\min}$ stands for the longest and shortest lengths of the **jobs**. Note that this can be easily seen to be bounded by n, the *number* of jobs.

Main Result

- Here we consider the following *restriction* of the `PartialResAll` problem:
  - The **multiplicity** version: resources are allowed to be picked multiple times.
  - Jobs have **unit** height.
- Call this problem the restricted `PartialResAll` problem.

Theorem

There is a polynomial time $O(\log \ell_{\text{max}}/\ell_{\text{min}})$-factor approximation for the restricted `PartialResAll` problem.

Here, ℓ_{max} and ℓ_{min} stands for the longest and shortest lengths of the **jobs**. Note that this can be easily seen to be bounded by n, the *number* of jobs.

Prior Work

- Recall that for the `ResAll` problem ($\{0, 1\}$ or multiplicity versions), a 4-factor approximation was known [BarNoy et al.01].
- Only special cases of the `PartialResAll` problem have been considered in the literature. The case where resources can be picked multiple times, and where jobs have arbitrary height, but unit length was considered in [Chakravarthy et al.11]. They show that in this special case, the `PartialResAll` problem has a 16-factor approximation.
- It is not known whether the $\{0, 1\}$ version of the above (partial covering) problem (i.e. jobs have unit length, resources can be picked at most once) has a constant factor approximation.

Prior Work

- Recall that for the `ResAll` problem ($\{0,1\}$ or multiplicity versions), a 4-factor approximation was known [BarNoy et al.01].
- Only special cases of the `PartialResAll` problem have been considered in the literature. The case where resources can be picked multiple times, and where jobs have arbitrary height, but unit length was considered in [Chakravarthy et al.11]. They show that in this special case, the `PartialResAll` problem has a 16-factor approximation.
- It is not known whether the $\{0,1\}$ version of the above (partial covering) problem (i.e. jobs have unit length, resources can be picked at most once) has a constant factor approximation.

Prior Work

- Recall that for the `ResAll` problem ($\{0, 1\}$ or multiplicity versions), a 4-factor approximation was known [BarNoy et al.01].
- Only special cases of the `PartialResAll` problem have been considered in the literature. The case where resources can be picked multiple times, and where jobs have arbitrary height, but unit length was considered in [Chakravarthy et al.11]. They show that in this special case, the `PartialResAll` problem has a 16-factor approximation.
- It is not known whether the $\{0, 1\}$ version of the above (partial covering) problem (i.e. jobs have unit length, resources can be picked at most once) has a constant factor approximation.

Warmup

- From now on, all jobs will have unit height, and resources can be picked multiple times.
- The basic intuition is that the case when jobs have unit lengths, it is easier to solve `PartialResAll` problems.
- So we would like to somehow reduce the problem to a situation where jobs have unit lengths.

Warmup

- From now on, all jobs will have unit height, and resources can be picked multiple times.
- The basic intuition is that the case when jobs have unit lengths, it is easier to solve `PartialResAll` problems.
- So we would like to somehow reduce the problem to a situation where jobs have unit lengths.

The overall theme

- We would like to simplify the structure of jobs.
- We can do a decomposition according to job lengths to reduce the general problem to the following special case, where the jobs form a so-called **mountain range**.
- This happens at a loss of $O(\log \ell_{\max}/\ell_{\min})$-factor.
- This is similar to the work of [Bansal et al.09] – they gave a logarithmic approximation for the UFP problem on a line.
- But how do we take care of the partiality parameter k?
- This happens via a DP.

The overall theme

- We would like to simplify the structure of jobs.
- We can do a decomposition according to job lengths to reduce the general problem to the following special case, where the jobs form a so-called **mountain range**.
- This happens at a loss of $O(\log \ell_{\max}/\ell_{\min})$-factor.
- This is similar to the work of [Bansal et al.09] – they gave a logarithmic approximation for the UFP problem on a line.
- But how do we take care of the partiality parameter k?
- This happens via a DP.

The overall theme

- We would like to simplify the structure of jobs.
- We can do a decomposition according to job lengths to reduce the general problem to the following special case, where the jobs form a so-called **mountain range**.
- This happens at a loss of $O(\log \ell_{\max}/\ell_{\min})$-factor.
- This is similar to the work of [Bansal et al.09] – they gave a logarithmic approximation for the UFP problem on a line.
- But how do we take care of the partiality parameter k?
- This happens via a DP.

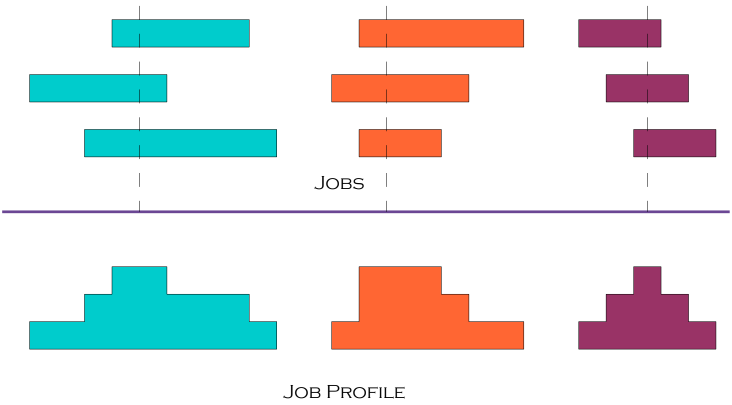
The overall theme

- We would like to simplify the structure of jobs.
- We can do a decomposition according to job lengths to reduce the general problem to the following special case, where the jobs form a so-called **mountain range**.
- This happens at a loss of $O(\log \ell_{\max}/\ell_{\min})$-factor.
- This is similar to the work of [Bansal et al.09] – they gave a logarithmic approximation for the UFP problem on a line.
- But how do we take care of the partiality parameter k?
- This happens via a DP.

The overall theme

- We would like to simplify the structure of jobs.
- We can do a decomposition according to job lengths to reduce the general problem to the following special case, where the jobs form a so-called **mountain range**.
- This happens at a loss of $O(\log \ell_{\max}/\ell_{\min})$-factor.
- This is similar to the work of [Bansal et al.09] – they gave a logarithmic approximation for the UFP problem on a line.
- But how do we take care of the partiality parameter k?
- This happens via a DP.

Example Mountain Range



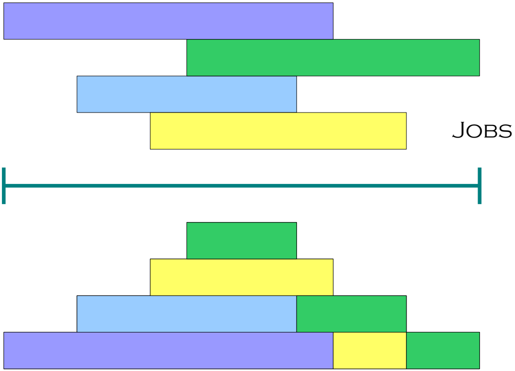
Mountain Ranges

- So we essentially have to prove that the special case of the problem where the jobs form a mountain range, has a constant factor approximation.
- Let's get down to an even more special case, where the mountain range consists of a single mountain.

Mountain Ranges

- So we essentially have to prove that the special case of the problem where the jobs form a mountain range, has a constant factor approximation.
- Let's get down to an even more special case, where the mountain range consists of a single mountain.

A Single Mountain

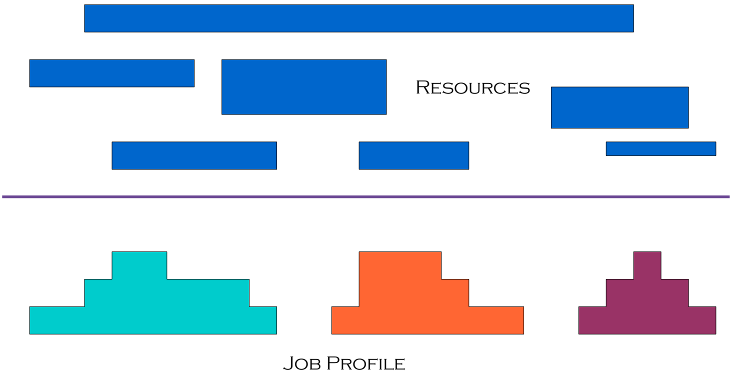


A SINGLE MOUNTAIN

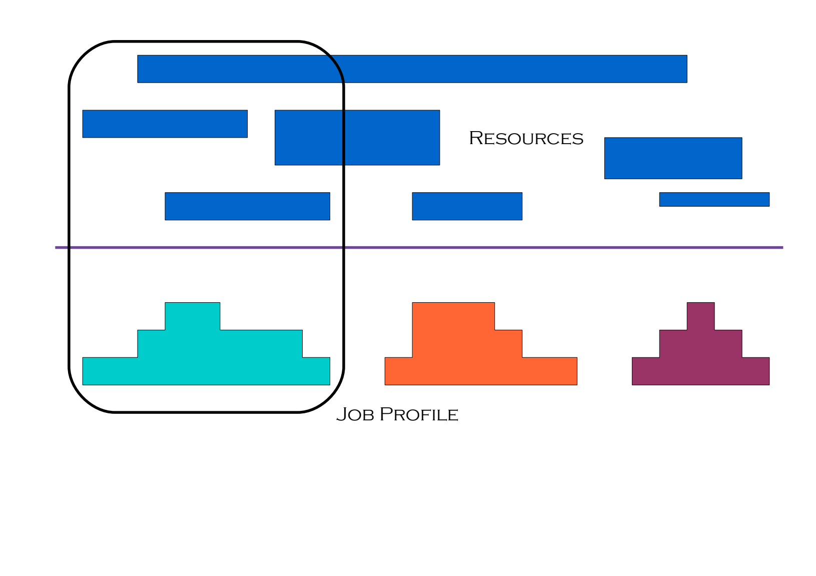
A Natural Algorithm for a Single Mountain

- Recall that all jobs have the same height. A natural idea is to *drop* jobs that extend farthest to the right (or to the left) while still observing the partiality constraint.
- This actually works: this yields a 8-factor approximation for the case of a single mountain.

Back to a mountain range



Back to a mountain range



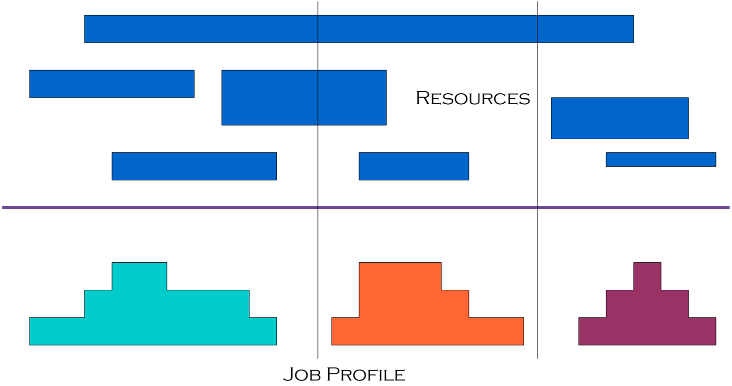
Back to a mountain range

- As of now, we have only decomposed *jobs*. How does a mountain range differ from a single mountain?
- Why does the problem concerning a mountain range not decompose into multiple single mountains?
- Because resources can span multiple single mountains of a mountain range.

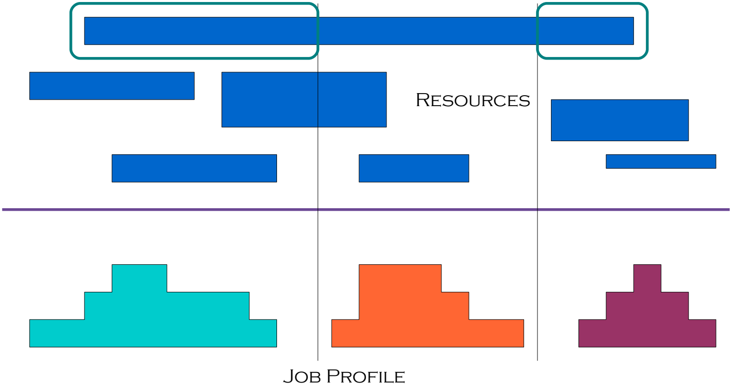
Back to a mountain range

- As of now, we have only decomposed *jobs*. How does a mountain range differ from a single mountain?
- Why does the problem concerning a mountain range not decompose into multiple single mountains?
- Because resources can span multiple single mountains of a mountain range.

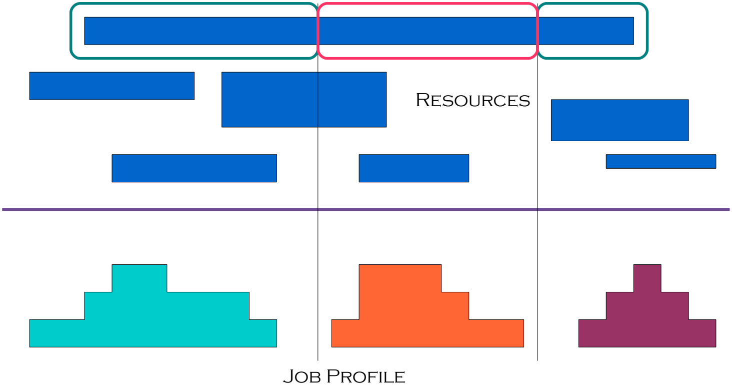
Back to a mountain range



Back to a mountain range



Back to a mountain range



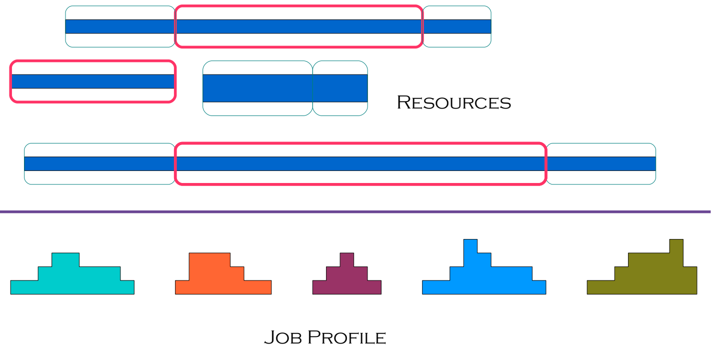
Resource Decomposition

- We decompose resources so that a resource fully spans a collection of (consecutive) single mountains (a long resource), or that a resource is contained in the span of only a single mountain (a short resource).
- This happens at a factor 3 loss.
- We can now “shrink" the mountains in a mountain range to timeslots, and thus we can modify our problem so that all the jobs are now one timeslot long.

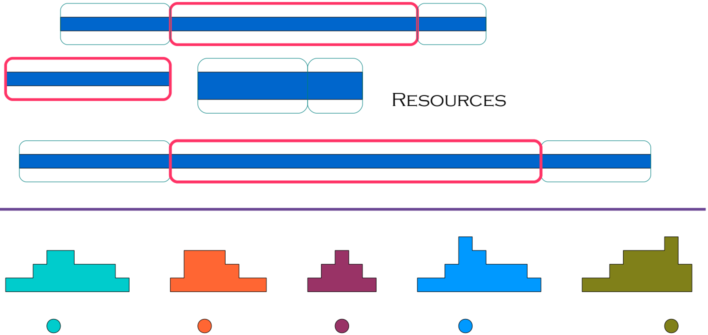
Resource Decomposition

- We decompose resources so that a resource fully spans a collection of (consecutive) single mountains (a long resource), or that a resource is contained in the span of only a single mountain (a short resource).
- This happens at a factor 3 loss.
- We can now “shrink" the mountains in a mountain range to timeslots, and thus we can modify our problem so that all the jobs are now one timeslot long.

Resource Decomposition



Resource Decomposition



MOUNTAINS SHRUNK TO TIMESLOTS WITH DEMAND = PEAK OF MOUNTAIN

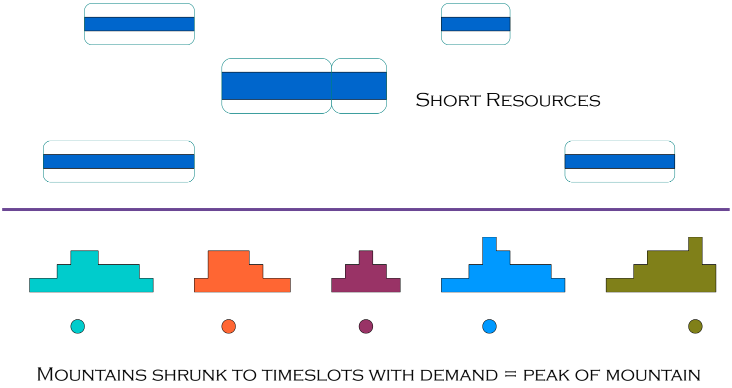
Resource Decomposition

MOUNTAINS SHRUNK TO TIMESLOTS WITH DEMAND = PEAK OF MOUNTAIN

Resource Decomposition

MOUNTAINS SHRUNK TO TIMESLOTS WITH DEMAND = PEAK OF MOUNTAIN

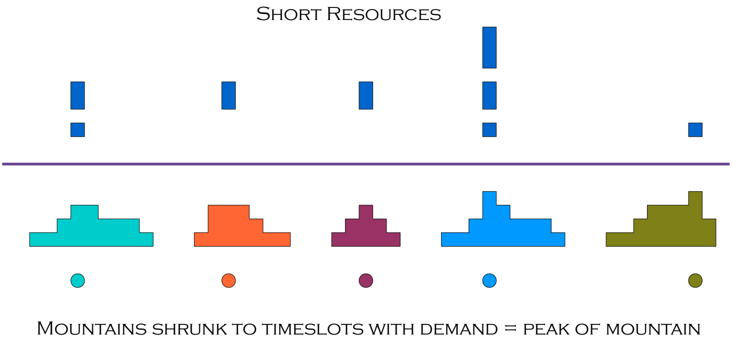
What happens to the short resources?



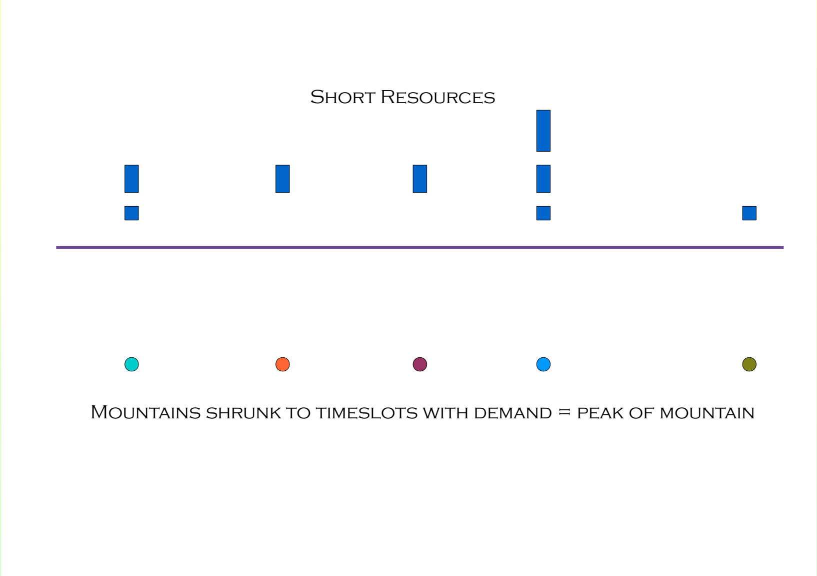
What happens to the short resources?

- The short resources along with the corresponding mountain form a “single mountain" `PartialResAll` problem.
- Solve this `PartialResAll` problem for various values of the partiality parameter k.
- For each (approximate) solution, place a resource of unit length on the corresponding timeslot of cost equal to that of the solution, and capacity equal to k.

What happens to the short resources?



What happens to the short resources?



The long and short of it. . .

MOUNTAINS SHRUNK TO TIMESLOTS WITH DEMAND = PEAK OF MOUNTAIN

Upshot

- Short resources comprise of a collection of (short) resources, so can be picked only once.
- Long resources are (parts) of original resources, so can be picked multiple times.
- What has happened to the partiality? We now need to pick up k of the cumulative demands d_t.

Upshot

- Short resources comprise of a collection of (short) resources, so can be picked only once.
- Long resources are (parts) of original resources, so can be picked multiple times.
- What has happened to the partiality? We now need to pick up k of the cumulative demands d_t.

Ruminations

- This is reminiscent of the $\{0, 1\}$ `PartialResAll` problem even when jobs are unit timeslot long.
- However, here the only resources restricted to be picked up at most once are the *short* resources.
- If only the long resources were present, we get a 16-factor approximation algorithm, via [Chakravarthy et al.11]. Why?
- If only the short resources are present, a simple DP gives the optimal solution.
- When both types of resources exist, a (slightly involved) amalgam of the DP solutions gives a 16-factor approximation for this "short resource, long resource" problem.

Ruminations

- This is reminiscent of the $\{0, 1\}$ `PartialResAll` problem even when jobs are unit timeslot long.
- However, here the only resources restricted to be picked up at most once are the *short* resources.
- If only the long resources were present, we get a 16-factor approximation algorithm, via [Chakravarthy et al.11]. Why?
- If only the short resources are present, a simple DP gives the optimal solution.
- When both types of resources exist, a (slightly involved) amalgam of the DP solutions gives a 16-factor approximation for this “short resource, long resource" problem.

Main Result

- This completes a proof sketch of the $\log(\ell_{\max}/\ell_{\min})$-factor approximation for the restricted case of `PartialResAll`.

Auxiliary Result

- One can also define the “Prize Collecting" version of `ResAll`. We prove a 4-factor approximation for the Prize Collecting version.
- However, this approximation algorithm does not have the so-called LMP (Lagrangian Multiplier Preserving) property (as in the framework of [JainVazirani99]).
- This is why, we could not appeal to [JainVazirani99] for the partial version.

Auxiliary Result

- One can also define the “Prize Collecting" version of `ResAll`. We prove a 4-factor approximation for the Prize Collecting version.
- However, this approximation algorithm does not have the so-called LMP (Lagrangian Multiplier Preserving) property (as in the framework of [JainVazirani99]).
- This is why, we could not appeal to [JainVazirani99] for the partial version.

Open Problems

- To prove a constant factor approximation for the $\{0, 1\}$ version of the `PartialResAll` problem.
- As a first step, prove this for the case where jobs are timeslots with demands (i.e. jobs have unit lengths).
- Remove the restriction of *unit heights* in the result discussed.
- Prove better hardness results. Strong NP-hardness?

THANK YOU.