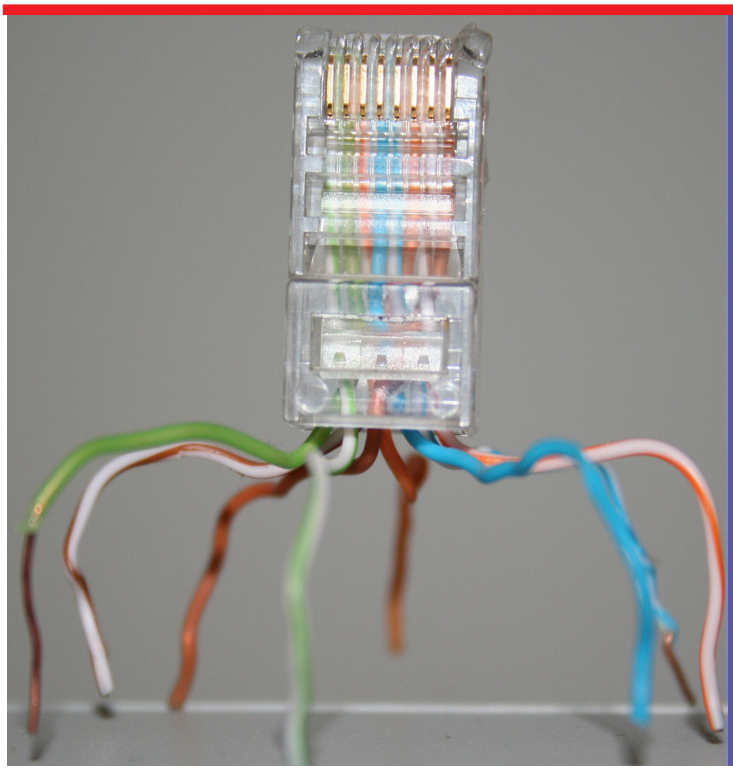
On the Integrated Job Scheduling and Constrained Network Routing Problem

Gamst, Mette

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

On the Integrated Job Scheduling and Constrained Network Routing Problem



Report 3.2010

DTU Management Engineering

Mette Gamst
January 2010

DTU Management Engineering
Department of Management Engineering

On The Integrated Job Scheduling and Constrained Network Routing Problem

M. Gamst
DTU Management, 2800 Kgs. Lyngby, Denmark
{gamst@man.dtu.dk}

This paper examines the problem of scheduling a number of jobs on a finite set of machines such that the overall profit of executed jobs is maximized. Each job demands a number of resources, which must be sent to the executing machine via constrained paths. A job cannot start before all its demand has arrived at the machine. Furthermore, two resource demand transmissions cannot use the same edge in the same time period. The problem has application in grid computing, where a number of geographically distributed machines work together for solving large problems. The machines are connected through an optical network.

The problem is formulated as a MIP problem and is shown to be $\mathcal{NP}$-hard. An exact solution approach based on Dantzig-Wolfe decomposition is proposed. Also, several heuristic methods are developed by combining heuristics for the job scheduling problem and for the constrained network routing problem.

The methods are computationally evaluated on test instances arising from telecommunications with up to 500 jobs and 500 machines. Results show that solving the integrated job scheduling and constrained network routing problem to optimality is very difficult. The exact solution approach performs better than using a standard MIP-solver; however, it is still unable to solve several instances. The proposed heuristics generally have good performance. Especially the First Come First Serve scheduling heuristic combined with a routing strategy, which proposes several good routes for each demand, has good performance with an average solution value gap of 3%. All heuristics have very small running times.

Key words: Job Scheduling; Network Routing; Routing and Wavelength Assignment; Grid Computing; Heuristics; Branch-and-Bound;

1 Introduction

Heuristic and exact solution methods for The Integrated Job Scheduling and Constrained Network Routing Problem (JSCNR) are presented. JSCNR consists of scheduling jobs on machines with respect to job demand transmission in an undirected network. The objective is to maximize the profit of scheduled jobs. It is assumed that the set of jobs, the set of machines, and the state of the network is known in advance; hence the problem can be viewed as being *offline*. Each job has a certain demand and a time window for execution. The demand must arrive at the machine before execution can begin. Each machine also has a time window and can execute at most one job at a time. Finally, the demand must be routed through an undirected network such that two demands do not share an edge in the same time slot. If the demand exceeds the capacity of an edge, then the demand transmission may occupy the edge in several time slots.

The problem has application in distributed production systems where a set of jobs can be carried out at various plants. If the total job execution exceeds the total amount of available machines and if the transportation paths are limited, it is necessary to consider both problems simultaneously. A typical application is the steel industry where the production can be placed at various sites, but the transportation of iron ore and coal by e.g. train constitutes a substantial logistic problem.

The problem also has application in grid computing where jobs are to be executed at various grid resources (machines) and where the grid resources are connected through an undirected optical

network. A job cannot be executed before its input data has arrived at the executing grid resource and two data transmissions cannot use the same wavelength on the same fiber at the same time.

An example is The Large Hadron Collider (LHC) Physics Program by The European Organization for Nuclear Research (CERN). It is estimated that the LHC experiments generate 15 petabytes of data annually [5], thus the project utilizes grid computing not only for distributing the scientific work, but also for distributing data storage. The network connections for the grid computing system must support high bandwidth availability, like e.g. optical networks. For details on the Worldwide LHC Computing Grid, see [5]. See Bates [3] for a thorough description of optical networks and its applications.

The contribution of this paper is to model and solve JSCNR. We show that the problem is $\mathcal{NP}$-hard and propose several heuristic and exact solution methods. The exact solution method is based on applying Dantzig-Wolfe decomposition such that the master problem determines where and when jobs are executed and the pricing problem calculates routing schemes. The heuristics are based on combining methods for The Integrated Job Scheduling and Network Routing Problem (JSNR) and for The Constrained Network Routing Problem (CNR).

Two types of test instances are generated: a tandem topology with 10-200 jobs and 10-500 machines and a real-life network topology taken from the Nordic DataGrid Facility with 10-200 jobs and 14 machines. The suggested solution methods are evaluated on the test instances. The exact solution method performs better than applying `CPLEX` on a MIP formulation; however, it is unable to solve several of the considered test instances within a half hour time frame. The heuristics are capable of solving all instances within minutes. Best general heuristic performance is reached when using the First Come First Serve strategy for JSNR and a routing scheme which suggests 2 different paths for each demand for CNR. This setting gives an average gap of 3%.

This paper is structured as follows. First JSCNR is defined in Section 2. Related work from the literature is also presented in this section along with notation and a mathematical model. In Section 3 heuristic methods are presented as combinations of methods for JSNR and for CNR. The heuristics are presented prior to the exact approach in Section 4, because they are used for solving the pricing problem and for finding a feasible start solution in the exact method. The suggested solution methods are computationally evaluated in Section 5 and final remarks are given in Section 6.

2 Problem definition

This section defines The Integrated Job Scheduling and Network Routing Problem (JSNR) and The Constrained Network Routing Problem (CNR). The problems are combined into The Integrated Job Scheduling and Constrained Network Routing Problem (JSCNR). For each problem an overview of work in the literature is given.

JSNR is closely related to JSCNR and only differs in the routing of job demands. Given is a set of jobs where each job has a certain demand, an estimated execution time, and a time window for execution. We also have a set of machines where each machine has an availability time window and can execute at most one job at time. Jobs must be assigned to machines and all job demand must arrive at the machine before execution can begin. The demand is routed through a capacitated network consisting of nodes and edges; the amount of demand on an edge in a time slot must not exceed the corresponding edge capacity. If the demand is larger than the edge capacity, then the demand can visit the edge in several time slots until all demand has been sent. The objective of the problem is to maximize the profit of executed jobs.

JSNR has application in production systems where transportation of goods from storage to production centers may constitute a logistical problem. The problem also has application in telecommunications; specifically in grid computing where jobs are executed on grid resources and where job input files must be sent to the executing grid resource through a (non-optical) network before execution can begin.

A simple version of JSNR was proved to be $\mathcal{NP}$-hard and greedy heuristics were presented by Marchal et al. [16].

An offline scheduler consisting of two steps was presented by Agarwal et al. [1]: first jobs were scheduled to grid resources such that the total penalty of delayed job executions was minimized, then the overall starting and end times of job schedules were determined.

Elghirani et al. [9] proposed a tabu search algorithm, which assigned jobs to a set of grid resources. The solution neighbourhood consisted of moving a scheduled job to another available grid resource and often used moves were penalized to avoid move cycles. When no improvement was reached in a certain time interval, the tabu list was cleared, a new random solution was found, and the tabu procedure started all over.

Varvaigos et al. [21] considered job routing and scheduling to support advance reservation. Advance reservation consists of reserving bandwidth and a grid resource for later execution of a given job. Varvaigos et al. considered one job and data transmission at a time; hence their algorithm can be viewed as being an online algorithm.

JSNR was shown to be $\mathcal{NP}$-hard and solved to optimality by Gamst and Pisinger [12]. The solution method was based on Dantzig-Wolfe decomposition where the pricing problem assigned a single job to a single machine, the branching strategy added cuts to strengthen the formulation, and the master problem found an overall feasible solution. Results showed that their branch-and-cut-and-price algorithm outperformed both simpler branch-and-price algorithms and `CPLEX`. The algorithm was capable of solving instances with up to 1000 jobs and 1000 machines within minutes.

The telecommunication application of JSNR was solved heuristically by Gamst [11] using a number of greedy heuristics, a swap-based metaheuristic and the adaptive large neighbourhood metaheuristic. Results showed that though the metaheuristics found better solution values than the greedy methods, they also had relatively large running times.

CNR consists of sending demand through a network such that two routes never use the same edge at the same time. Given is a network consisting of nodes and capacitated edges. The network takes time into account, i.e., an edge can be visited at different time slots. Also given is a set of routing requests each consisting of a source, a destination, a routing time window, and an amount of demand. To satisfy a routing request, the demand must be sent from the source to the destination within the time window. If the amount of demand exceeds an edge capacity, then it takes several time slots to route the demand on that edge. Two routes cannot use the same edge at the same time.

CNR has application in the transportation sector. When routing trains through a railway infrastructure, two trains cannot use the same section of railway tracks at the same time. Also, the length of the train determines how long it takes to travel across a stretch of railway tracks. Each train has some starting and ending point and the goods on the train must arrive before a certain time.

CNR also has application in telecommunications where it corresponds to the $\mathcal{NP}$-hard static Routing and Wavelength Assignment Problem (RWA). The problem is to establish a number of connections (or light paths) in an optical network such that each connection travels from its source to its destination in a certain time window using one or more wavelengths. Two connections cannot use the same wavelength on the same fiber at the same time. The RWA is static since we have full knowledge on the problem instance in advance.

Most work on the RWA in the literature focuses on maximizing the number of established data connections. The underlying optical network is typically considered to be one of three topologies: wavelengths cannot be converted, see Zang et al. [22], wavelengths can be converted in all nodes, see Ramamurthy and Mukherjee [17], and wavelengths can be converted in a subset of nodes, see Iness and Mukherjee [14]. The RWA was proved $\mathcal{NP}$-hard by Chlamtac et al. [6].

The RWA problem is typically solved using a heuristic decomposition which consists of a routing problem and a wavelength assignment problem. The routing problem suggests one or more paths for each data connection. The wavelength assignment problem finds an available wavelength and

assigns it to one of the proposed paths for each data connection. An overview of heuristics from the literature for solving the decomposed RWA is presented by Zang et al. [22].

JSCNR consists of combining JSNR and CNR: jobs must be assigned to machines such that all job demand arrives at the machine before execution begins. The job demand is routed through an undirected network such that two routes never travel on the same edge at the same time. The network topology connects edges in such a way that the corresponding RWA does not support wavelength conversion. Jobs must be assigned to machines for execution such that the total profit of executed jobs is maximized. JSCNR is $\mathcal{NP}$-hard as it contains both the $\mathcal{NP}$-hard JSNR and the $\mathcal{NP}$-hard CNR as special cases.

2.1 Mathematical Formulation

Notation from applying JSCNR in a telecommunications context is used in the following formalization. This means that we consider the problem of assigning jobs to resources where job data must be routed through an optical network. The optical network is dedicated to the job scheduling process, hence paths between all terminal nodes are known in advance.

The set of jobs is denoted J, the set of resources is R, the set of edges is E and the set of time stamps is T. Note that time is discrete, i.e., is given in time stamps $t \in T$.

The set of wavelengths on edge $(i,k) \in E$ is denoted λ_{ik} and the set of all wavelengths is denoted λ. For a wavelength $l \in \lambda$ let E_l denote the set of edges which are capable of carrying data on wavelength l. All wavelengths on all edges have the same bandwidth capacity d.

Let t_{λ^+} denote the time it takes to establish a new wavelength on an edge and let t_{λ^-} denote the time it takes to release a wavelength on an edge. The reason for introducing these time buffers is to make the solution more robust: if a data transmission is delayed, then it will not be interfered by a new transmission if the delay is less than t_{λ^-}. Furthermore, a data transmission does not start until t_{λ^+} time after the wavelength is assigned thus leaving even further room for the previous transmission to finish. Introducing these extra time buffers has a drawback; the extra time buffers may prevent more jobs to be executed. When solving the problem, the grid administrator should thus experiment with the size of the time buffers in order to reach an appropriate trade-off between robustness and job execution.

Each job $j \in J$ is assigned a time window $[a_j, b_j]$, the estimated computation time Q_j, the total size of the job data S_j, the amount of data p_j^r placed on each resource $r \in R$, and a profit $c_j \in \mathbb{R}^+$ for execution.

Each resource $r \in R$ is assigned an availability start time a_r and end time b_r. To simplify notation, the time window $[a_{ik}, b_{ik}]$ is introduced, where $a_{ik} = \max\{a_i, a_k\}$ and $b_{ik} = \min\{b_i, b_k\}$ for $i, k \in R \cup J$. For further notational convenience, two sets are introduced: J_t and R_t. The set J_t consists of jobs j with $a_j \leq t \leq b_j$. Similarly, the set R_t consists of resources r with $a_r \leq t \leq b_r$.

Now, the mathematical model includes two types of variables $x_j^{tr} \in \{0,1\}$ and $x_{ikl}^{tj} \in \{0,1\}$. If $x_j^{tr} = 1$ then job $j \in J$ is executed on resource $r \in R$ with execution beginning at time $t \in T$. If $x_j^{tr} = 0$ then the job is not executed on the resource with this beginning time. If $x_{irl}^{tjk} = 1$ then edge $(i,r) \in E$ is carrying data original stored on resource $k \in R$ on wavelength $l \in \lambda_{ir}$ at time $t \in T$ for job $j \in J$. Otherwise, $x_{irl}^{tjk} = 0$. JSCNR is formulated as:

$$\max \quad \sum_{r\in R}\sum_{j\in J}\sum_{t=a_{rj}}^{b_{rj}-Q_j} c_j x_j^{rt} \tag{1}$$

$$\text{s. t.} \quad \sum_{r\in R}\sum_{t=a_{rj}}^{b_{rj}-Q_j} x_j^{rt} \le 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{i\in R}\sum_{t'=a_{ri}}^{t-1}\sum_{l\in\lambda_{ri}} x_{ril}^{t'jr} \ge \left\lceil \frac{p_r^j}{d} \right\rceil \sum_{i\in R\setminus\{r\}} x_j^{it} \qquad \forall r\in R, \forall j\in J, \forall t\in[a_{rj}, b_{rj}-Q_j] \tag{3}$$

$$\sum_{i\in R}\sum_{k\in R\setminus\{r\}}\sum_{t'=a_{ir}}^{t-1}\sum_{l\in\lambda_{ir}} x_{irl}^{t'jk} \ge \left\lceil \frac{S_j - p_r^j}{d} \right\rceil x_j^{rt} \qquad \forall r\in R, \forall j\in J, \forall t\in[a_{rj}, b_{rj}-Q_j] \tag{4}$$

$$\sum_{t'=t+1}^{b_{rj}-Q_j} x_j^{rt'} = 0 \Rightarrow \quad \sum_{i\in R_t:(i,r)\in E_l} x_{irl}^{tjk} - \sum_{i\in R_t:(r,i)\in E_l} x_{ril}^{(t+1)jk} = 0 \qquad \forall k, r\in R: p_k^j > 0, \forall j\in J, \tag{5}$$

$$\forall l\in\lambda, \forall t\in[a_r, b_r]$$

$$\sum_{k\in R: p_k^j>0}\sum_{j\in J_t}\sum_{t'=t}^{t+t_{\lambda^-}+t_{\lambda^+}} (x_{irl}^{t'jk} + x_{ril}^{t'jk}) \le 1 \qquad \forall (i,r)\in E, \forall l\in\lambda_{ir}, \forall t\in[a_{ir}, b_{ir}] \tag{6}$$

$$\sum_{j'\in J\setminus j}\sum_{t'=t}^{\min\{t+Q_j, b_{j'r}-Q_{j'}\}} x_{j'}^{rt'} + Q_j x_j^{rt} \le Q_j \qquad \forall j\in J,\ r\in R, t\in[a_{jr}, b_{jr}-Q_j] \tag{7}$$

$$x_j^{rt} \in \{0,1\} \qquad \forall j\in J, \forall r\in R, \forall t\in[a_{rj}, b_{rj}] \tag{8}$$

$$x_{irl}^{tjk} \in \{0,1\} \qquad \forall j\in J, \forall k\in R, \forall (i,r)\in E, \quad \forall l\in\lambda_{ir}, \forall t\in[a_{rj}, b_{rj}] \tag{9}$$

The objective (1) maximizes the profit of executed jobs. The first constraint (2) says that each job can be executed at most once. If a job is executed on some resource $i \in R$ then data from all other resources $r \in R$ must be sent out on the network (3). Constraint (4) says that if a job is executed at resource $r \in R$ then all data must arrive before execution time. Flow conservation is ensured in (5). Data arriving at some node at time t must leave the node again at time $t+1$ unless the job is executed at this node. Constraint (6) forbids several paths from using the same wavelength on the same edge at the same time. Finally, the last constraint (7) says that a resource can execute at most one job at a time. Bounds ensure that variables take on feasible values.

3 Greedy heuristic solution approach

In this paper, we consider the heuristic approach for The Integrated Job Scheduling and Constrained Network Routing Problem (JSCNR), which combines greedy heuristics for The Integrated Job Scheduling and Network Routing Problem (JSNR) and for The Constrained Network Routing Problem (CNR). JSNR was solved heuristically by Gamst [11]. The data transmission part of the heuristics, however, must be replaced by algorithms for the CNR. The latter has application in telecommunications as the Routing and Wavelength Assignment Problem (RWA) problem for which many solution methods are presented in the literature see e.g. the survey of Zang et al. [22].

Let us first consider heuristics for JSNR in the literature (see Gamst and Pisinger [11] or Sørensen [19] for more details):

- First Come First Serve. The first job on queue is assigned to the resource at which execution finishes first. Let $|\mathrm{T}_{\mathrm{data}}|$ denote the running time of transmitting data. The theoretical running time for the First Come First Serve heuristic is $\mathcal{O}(|J||R||\mathrm{T}_{\mathrm{data}}|)$, since the heuristic in worst case attempts to assign each job to all resources.

- Best First. The job with highest profit is assigned to the resource at which job execution finishes first. The running time is $\mathcal{O}(|J|\log|J| + |J||R||\mathrm{T_{data}}|)$ where $|\mathrm{T_{data}}|$ is the running time for the data transmission problem, since jobs first are sorted according to profit and then the heuristic in worst case tries to assign each job to all resources.

- First Fit. For each resource, the job with earliest execution finish time is executed. If a draw between several jobs are reached then the job with highest profit is selected. The theoretical running time is $\mathcal{O}(|R||J|^2|\mathrm{T_{data}}|)$ where $|\mathrm{T_{data}}|$ is the running time for the data transmission problem, because for each resource the heuristic assigns all pairs of jobs in order to compare the execution finish times.

- Random Fit. Randomly selected jobs are assigned to each resource. The running time is $\mathcal{O}(|J||R||\mathrm{T_{data}}|)$ where $|\mathrm{T_{data}}|$ is the data transmission running time, because in worst case the heuristic tries to assign each job to all resources.

These four heuristics need to know how long it takes to transmit job data to a resource in order to determine execution start and end times. The time it takes to transmit job data is found by solving the CNR problem heuristically.

CNR is solved as the RWA and we propose using a subset of the heuristics for the RWA in the literature. When solving the RWA as part of the JSCNR, the RWA may be solved a large number of times. Thus if the heuristic for the RWA has high complexity, then the overall solution procedure will suffer. The selected heuristics have relatively small running times and all divide the RWA into a routing problem and a wavelength assignment problem. The selected heuristics for the routing problem are:

- Fixed-Alternate Routing. Several paths are found for each data connection request; see Banerjee et al. or Birman and Kershenbaum [2, 4]. The heuristic corresponds to the k-shortest path problem, when the number of generated paths for the data connection corresponds to k. Thus the theoretical running time for establishing a single data connection equals that of the k-shortest path problem; $\mathcal{O}(|E| + |V|\log|V| + k)$ where $|V|$ is the number of nodes in the network, see Eppstein [10].

- Adaptive Routing. This method runs a shortest path algorithm on the graph where edge costs are based on previously chosen routes; see Zang et al. [22]. The theoretical running time for establishing a single data connection corresponds to the running time for a shortest path algorithm, e.g., $\mathcal{O}((|E| + |V|)\log|V|)$ which is the running time of Dijkstra's algorithm using a binary heap, see Cormen et al. [7].

The selected heuristics for wavelength assignment are:

- First Fit. The first available wavelength is assigned to the current data connection request; see Birman and Kershenbaum or Kovacevic and Acampora [4, 15]. The running time for assigning a wavelength to a single data connection is $\mathcal{O}(|\lambda||E|)$ where $|\lambda|$ is the number of wavelengths, as the heuristic in worst case investigates the availability of each wavelength on all edges.

- Most Used. Among the available wavelengths for a data connection request, the wavelength which so far has been used the most is assigned to the data connection request, see Subrarnaniam and Barry [20]. The theoretical running time is $\mathcal{O}(|\lambda|\log|\lambda| + |\lambda||E|)$, because first the availability of all wavelengths on all edges is found, then the wavelengths are sorted according to usage, and finally the heuristic investigates the availability of each wavelength from the sorted list on all edges.

- Random Assignment. An available wavelength is randomly selected and assigned to the current data connection request. Running time is $\mathcal{O}(|\lambda||E|)$ where $|\lambda|$ is the number of wavelengths, because in worst case the heuristic investigates the availability of each wavelength on all edges.

3.1 Heuristics for JSCNR

Combining the heuristics from the previous section results in heuristics for JSCNR. The heuristics are displayed in the following tables along with their theoretical running times. The first table uses Fixed-Alternate routing, the second Adaptive routing. The first row in each part of the tables consists of the name of the JSNR heuristic. The remaining three rows in each part of the tables consist of the name of the wavelength assignment heuristics and the corresponding theoretical running time for combining the JSNR and CNR heuristics:

FCFS	**Fixed-alternate**
First fit	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + k + \|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + k + \|\lambda\|(\log\|\lambda\| + \|E\|)))$
Random	$\mathcal{O}(\|J\|\|R\|\|(\|V\|\log\|V\| + k + \|\lambda\|\|E\|))$
Best first	
First fit	$\mathcal{O}(\|J\|(\log\|J\| + \|R\|\|(\|V\|\log\|V\| + k + \|\lambda\|\|E\|)))$
Most used	$\mathcal{O}(\|J\|(\log\|J\| + \|R\|(\|V\|\log\|V\| + k + \|\lambda\|(\log\|\lambda\| + \|E\|))))$
Random	$\mathcal{O}(\|J\|(\log\|J\| + \|R\|(\|V\|\log\|V\| + k) + \|\lambda\|\|E\|))$
First fit	
First fit	$\mathcal{O}(\|J\|^2\|R\|(\|V\|\log\|V\| + k + \|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|^2\|R\|(\|V\|\log\|V\| + k + \|\lambda\|(\log\|\lambda\| + \|E\|)))$
Random	$\mathcal{O}(\|J\|^2\|R\|(\|V\|\log\|V\| + k + \|\lambda\|\|E\|))$
Random fit	
First fit	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + \|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + \|\lambda\|(\log\|\lambda\| + \|E\|)))$
Random	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + \|\lambda\|\|E\|))$

FCFS	**Adaptive**
First fit	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + \|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + \|\lambda\|(\log\|\lambda\| + \|E\|)))$
Random	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + \|\lambda\|\|E\|))$
Best first	
First fit	$\mathcal{O}(\|J\|(\log\|J\| + \|R\|(\|V\|\log\|V\| + \|\lambda\|\|E\|)))$
Most used	$\mathcal{O}(\|J\|(\log\|J\| + \|R\|(\|V\|\log\|V\| + \|\lambda\|(\log\|\lambda\| + \|E\|))))$
Random	$\mathcal{O}(\|J\|(\log\|J\| + \|R\|(\|V\|\log\|V\| + \|\lambda\|\|E\|)))$
First fit	
First fit	$\mathcal{O}(\|J\|^2\|R\|(\|V\|\log\|V\| + \|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|^2\|R\|(\|V\|\log\|V\| + \|\lambda\|(\log\|\lambda\| + \|E\|)))$
Random	$\mathcal{O}(\|J\|^2\|R\|(\|V\|\log\|V\| + \|\lambda\|\|E\|))$
Random fit	
First fit	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + \|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + \|\lambda\|(\log\|\lambda\| + \|E\|)))$
Random	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\| + \|\lambda\|\|E\|))$

The theoretical running times in the tables are used for comparison with practical running times when computationally evaluating the heuristics in Section 5.

4 Exact Solution Approach

The exact solution approach is based on Dantzig-Wolfe decomposing The Integrated Job Scheduling and Constrained Network Routing Problem (JSCNR) such that the master problem decides where and when to execute jobs according to data transmission. The pricing problem decides when to send all data for each job according to the reduced costs. Recall the mathematical formulation (1)-(9). The master problem includes constraints (2), (6), and (7) and the pricing problem takes care of the remaining constraints along with (6).

Let the decision variable $y_p^{jrt} \in \{0, 1\}$ indicate if job j is executed on resource r at time t where job data is sent according to p. The pricing problem generates ways of sending data $p \in P$ for a given job, resource and execution time according to the reduced cost of the current solution. The

master problem is:

$$\max \quad \sum_{j\in J}\sum_{r\in R}\sum_{t=a_{rj}}^{b_{rj}-Q_j}\sum_{p\in P} c_j y_p^{jrt} \tag{10}$$

$$\text{s. t.} \quad \sum_{r\in R}\sum_{t=a_{rj}}^{b_{rj}-Q_j}\sum_{p\in P} y_p^{jrt} \le 1 \qquad \forall j\in J \tag{11}$$

$$\sum_{j\in J_t}\sum_{u\in R_t}\sum_{t'=t}^{t+t_{\lambda^-}+t_{\lambda^+}}\sum_{p\in P}(\delta_p^{irl}y_p^{jut'}+\delta_p^{ril}y_p^{jut'}) \le 1 \qquad \forall (i,r)\in E, \forall l\in\lambda_{ir}, \forall t\in[a_{ir},b_{ir}] \tag{12}$$

$$\sum_{j'\in J\setminus j}\sum_{t'=t}^{\min\{t+Q_j,b_{jr}-Q_j\}}\sum_{p\in P} y_p^{j'rt'} + Q_j\sum_{p\in P} y_p^{jrt} \le Q_j \qquad \forall j\in J,\ \forall r\in R, t\in[a_{jr},b_{jr}-Q_j] \tag{13}$$

$$y_p^{jrt}\in\{0,1\} \qquad \forall j\in J,\ r\in R, t\in[a_{jr},b_{jr}-Q_j] \tag{14}$$

The objective (10) maximizes the profit of executed jobs. The first constraint (11) ensures that a job can be executed at most once and the second constraint (12) ensures that each wavelength on each edge is visited by at most one data connection. Finally, constraint (13) says that a resource can execute at most one job at a time and the bound (14) forces variables to take on feasible values.

4.1 Pricing problem

The dual variables of the master problem are $\pi_j \ge 0$, $\omega_{irlt} \ge 0$ and $\rho_{jrt} \ge 0$ for constraints (11), (12), and (13), respectively. The reduced cost for a given job j, resource r and execution time t is:

$$c_j - \pi_j - Q_j\rho_{jrt} - \sum_{j'\in J\setminus\{j\}}\sum_{t'=\max\{t-Q_{j'}+1,a_{j'r}\}}^{\min\{t,b_{rj'}-Q_{j'}\}}\rho_{j'rt'} > \sum_{(i,r)\in E}\sum_{l\in L}\sum_{t'=a_{ir}}^{b_{ir}}\left(\omega_{irl}^{t'}+\omega_{ril}^{t'}\right) \tag{15}$$

When solving the pricing problem for a given job j, resource r, and execution time t we wish to minimize the right hand side of the reduced cost, because the value of the left hand side is already known. Hence the pricing problem is to find a way of sending all job data for job j to resource r in time for job execution at time t such that the right hand side of (15) is minimized.

The decision variable $y_{iul}^{t'k}\in\{0,1\}$ is introduced to indicate data transmission in the pricing problem. Let $y_{iul}^{t'k}$ denote whether or not data stored on resource $k\in R$ is travelling on edge $(i,u)\in E$, using wavelength $l\in\lambda$ at time $t'\in[a_{ir},t[$. The pricing problem is:

$$\min \quad \sum_{k\in R}\sum_{(i,u)\in E}\sum_{l\in L}\sum_{t'\in[a_{ir},t[}\left(\omega_{iul}^{t'}y_{iul}^{t'k}+\omega_{uil}^{t'}y_{uil}^{t'k}\right) \tag{16}$$

$$\text{s. t.} \quad \sum_{i\in R}\sum_{t'=a_{ki}}^{t-1}\sum_{l\in\lambda_{ki}} y_{kil}^{t'k} \ge \lceil\frac{p_k^j}{d}\rceil \qquad \forall k\in R \tag{17}$$

$$\sum_{k\in R:p_k^j>0}\sum_{(i,r)\in E}\sum_{t'=a_{ir}}^{t-1}\sum_{l\in\lambda_{ir}} y_{irl}^{t'k} \ge \lceil\frac{S_j-p_r^j}{d}\rceil \tag{18}$$

$$\sum_{i\in R_t:(i,u)\in E_l,u\neq r} y_{iul}^{t'k} - \sum_{i\in R_t:(u,i)\in E_l,u\neq r} y_{uil}^{(t'+1)k} = 0 \qquad \forall u,k\in R\setminus\{r\}:p_k^j>0,\ \forall l\in\lambda,\forall t'\in[a_u,t[\tag{19}$$

$$\sum_{j\in J_t}\sum_{k\in R:p_k^j>0}\sum_{u\in R_t}\sum_{t''=t'}^{t'+t_{\lambda^-}+t_{\lambda^+}}(y_{iul}^{t''k}+y_{uil}^{t''k}) \le 1 \qquad \forall (i,u)\in E,\forall l\in\lambda_{ir},\forall t'\in[a_{iu},t[\tag{20}$$

$$y_{irl}^{t'k}\in\{0,1\} \qquad \forall k\in R:p_k^j>0,\ \forall (i,r)\in E,\ \forall l\in\lambda,\forall t'\in[a_{ir},t[\tag{21}$$

The objective (16) minimizes the right hand side of (15). The first constraint (17) says that all job data must be sent from each data source. The next constraint (18) makes sure that all job data arrives at the executing resource r before job execution time t. Constraint (19) ensures flow conservation. Finally constraint (20) says that no more than one data connection can use a wavelength on an edge at a time and the bound (21) forces variables to take on feasible values.

The pricing problem is the Routing and Wavelength Assignment Problem (RWA) over time and is $\mathcal{NP}$-hard. Hence we try to generate columns heuristically and only solve the pricing problem to optimality when no heuristic columns with positive reduced cost can be found. The proposed greedy heuristics for the RWA in Section 3 are applied on the pricing problem to generate columns heuristically. The heuristics are modified slightly: when they can choose between several paths or wavelengths, then the cheapest option according to (16) is selected.

The exact solution approach is based on solving the mathematical formulation for the RWA problem over time. Recall that all paths between all pairs of resources are known in advance. In the mathematical formulation we generate a column for each path at each possible start time using each wavelength. The exact solution approach is solved for a given job j, an executing resource r, and an execution time t. Let $\mathcal{P}$ denote the set of columns. The variable $y_p \in \{0,1\}$ indicates whether or not column $p \in \mathcal{P}$ is included in the current solution. Three constants are introduced: $\delta_p^{iklt'}$ denotes whether or not column p uses wavelength $l \in \lambda$ on edge $(i,k) \in E$ at time $t' \in [a_{ikj}, t[$, δ_p^k denotes whether or not column p routes data stored at resources k, and c_p denotes the reduced cost for column p. The model is:

$$\min \quad \sum_{p\in\mathcal{P}} \sum_{(i,k)\in E} \sum_{l\in\lambda} \sum_{t'\in[a_{ikj},t[} \delta_p^{iklt'} c_p y_p \tag{22}$$

$$\text{s. t.} \quad \sum_{p\in\mathcal{P}} \delta_p^{iklt'} y_p + \delta_p^{kilt'} y_p \leq 1 \qquad \forall (i,k) \in E, \forall l \in \lambda, \forall t' \in [a_{ikj}, t[\tag{23}$$

$$\sum_{p\in\mathcal{P}} \delta_p^k y_p = 1 \qquad \forall k \in R : p_k^j > 0 \tag{24}$$

$$y_p \in \{0,1\} \qquad \forall p \in \mathcal{P} \tag{25}$$

The objective function (22) minimizes the reduced cost. The first constraint (23) says that each wavelength on each edge can be used at most once and constraint (24) ensures that all data connections are established exactly once.

The number of columns in (22)-(25) is polynomial in the input size: the path between two terminal nodes is known in advance. We must decide when to travel on the path, thus we generate a path variable for each path at each available travel time and for each wavelength. Let $\mathcal{O}(|K|)$ be the number of data connections, $\mathcal{O}(|T|)$ be the number of available travel times, and $\mathcal{O}(|\lambda|)$ be the number of wavelengths; the number of variables is $\mathcal{O}(|\lambda||T||K|)$.

4.2 Branching strategy

Branching ensures that variables in the LP-relaxed master problem eventually take on binary values. To determine the branching strategy we investigate when variable values may become fractional:

1. A job is only partially executed
2. A job is executed on the same resource but at different times
3. A job is executed on different resources
4. A job is executed on a given resource at a given time using routing times which differ in the latest data arrival time
5. A job is executed on a given resource at a given time using routing times which differ in the used wavelengths

In the first case we generate two branching children in each of which we add the constraint:

$$\sum_{p \in P} \delta_p^j y_p = 0 \quad \text{vs.} \quad \sum_{p \in P} \delta_p^j y_p = 1 \tag{26}$$

which ensures that job j is either not executed or it is fully executed. The branching constaint adds a dual variable ω_j, which the pricing problem must handle. Because the pricing problem is solved for each job, the extra dual variable can be added to the left hand side of (15) and does not interfere with the pricing problem.

The second case is handled by finding a time stamp lying between the current execution times. Two branching children are generated: in the first child the job must be executed no later than the time stamp and in the second child the job must be executed no earlier than the time stamp. In each child, columns with illegal execution times are set to zero. The pricing problem is altered slightly into setting bounds on execution times and not allowing data to arrive later than the latest execution start time.

The third case is handled by choosing a resource on which the job is partially executed. Two branching children are generated: in the first child the job must be executed on the resource, and in the second child the job cannot be executed on the resource. In each branching child, columns using an illegal executing resource are set to zero. The pricing problem is modified slightly into either forcing execution on a certain resource or to not allowing execution on illegal resources.

In the fourth case the data transmission times and possibly the used wavelengths differ. The case is handled by finding a time stamp for routing. Two branching children are generated: in the first child all data must arrive before the time stamp and in the second child all data cannot be sent before the time stamp. In each child the variables with illegal routing times are set to zero. The pricing problem is modified into not allowing routing at illegal times by excluding predefined columns using illegal routing times.

In the fifth case the execution and data transmission times are equal for all non-zero variables. Only the used wavelengths differ. The case is handled by choosing a wavelength for a data transmission path. The chosen wavelength must be used by at least one of the fractional variables in the current solution. Two branching children are generated: in the first child the chosen wavelength must be used on the chosen path, thus all variables which use different wavelengths are set to zero. In the second branching child the chosen wavelength cannot be used on the chosen path, thus all variables which use the chosen wavelength are set to zero. The pricing problem is modified into including or excluding columns using the chosen wavelength on the chosen path, respectively.

4.3 Start solution

The master problem must initially hold one or more columns before values for dual variables can be found for the pricing problem. To reach a start solution we can apply the greedy heuristics from Section 3 on the problem instance. The heuristics, however, do not guarantee to find a feasible solution even if one exists. In this case, an exact solution approach must try to assign a job to a resource. We choose to run a modified version of the exact solution approach for the pricing problem; instead of minimizing the reduced cost, the exact approach only decides whether or not it is possible to assign a given job to a given resource.

4.4 Reducing the number of constraints

The master problem consists of a large number of constraints, especially as time window sizes increase. Some instances may not utilize large parts of the time windows; hence it would be beneficial to leave out constraints for unused time stamps. Through preliminary results we have noted a significant improvement of approximately 35% on time usage when including all constraints of type (11) and only violated constraints of type (12) - (13). Separation routines for identifying violated constraints consider all non-negative variables for all possible constraints and thus have polynomial running time in the input size.

Including only violated constraints does not impose any changes on neither the pricing problem nor the branching strategies. When calculating the reduced costs, only dual variables for constraints included in the master problem are considered.

4.5 Reducing the number of iterations

Preliminary results have shown that the branch-and-cut-and-price algorithm runs through a relatively large number of iterations before finding a lower bound in a search tree node. The reason for this may be that the dual variables take on inappropriate values, hence the algorithm prices in many unused columns before finally converging toward the lower bound. A way to avoid this is by applying a method for stabilizing the values of dual variables. Several stabilization methods are presented in the literature. They typically consist of setting bounds on how much the values of the dual variables may change from one iteration to the next. The bounds may be in the form of boxes for each dual variable, see Rousseau et al. [18] or by adding a punishment in the objective function for the distance between the former and the current value of each dual variable, see DuMerle et al. [8]. Rousseau et al. [18] suggest an interior-point stabilization method where the values of dual variables are set to a linear combination of extreme points in the dual solution space. The stabilization method can easily be applied to the master problem by changing the bounds on constraints and variables whose corresponding dual variables and constraints are not tight. For details, see Rousseau et al. [18] who show how to apply the stabilization method on the Set Cover problem. We have applied the interior-point stabilization method and preliminary results show that the method decreases time usage with up to 67%.

5 Computational experiments

The proposed solution methods are tested. In this section we first introduce the generated problem instances, then a computational evaluation of the proposed exact method and heuristics for JSCNR is presented.

5.1 Test instances

Two types of problem instances are generated. Both instance types arise in telecommunications and are denoted the NDGF and the Tandem instances, respectively.

NDGF
A set of instances is based on the network topology of the Nordic DataGrid Facility (NDGF), which consists of a grid computing system in Scandinavia. Current projects on the NDGF include handling data from the Large Hadron Collider (LHC) by the European Organization for Nuclear Research (CERN), see [5]. The NDGF network topology was presented by Grønager [13] and consists of 14 nodes, which are connected in a sparse graph. An illustration can be seen in Figure 1. All data arrives from Europe to a grid resource in Denmark, which thus works as job data storage for all jobs. In three of the Scandinavian countries, grid resources are connected through a network node. These are marked as squares in Figure 1. The grid resource in Denmark and all network hubs are available at all times.

Tandem
A set of instances based on a *tandem* topology is generated. An example of a tandem network is given in Figure 2. All nodes but two are connected with exactly two other nodes. The two nodes in each end of the network are only connected with one other node. Hence, the number of edges in the test instances is always $|E| = 2(|V| - 1)$. This set of instances is introduced in order to test how larger networks are handled. The number of edges and nodes thus vary from instance to instance.

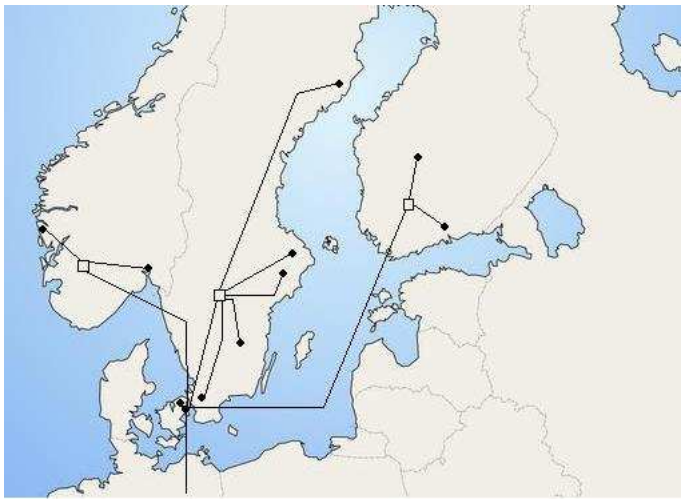


Figure 1: An illustration of the NDGF network. Resources are marked as filled circles, while the squares indicate nodes unable to execute jobs.

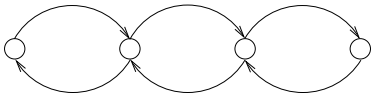


Figure 2: An example of a tandem network. Every node is only connected to its neighboring nodes.

Grid activity
The number of jobs, the number of wavelengths, and the amount of available bandwidth per wavelength vary from instance to instance. The size and distribution of job input files, the execution time, and the time window for each job are randomly generated. Similarly, the resource time windows are also randomly generated.

5.2 Results

The solution methods have been implemented in `C++` and tested on a 2.66 GHz Intel Xeon machine with 16 GB RAM. Note that CPU times in the following stem from using one core. All test runs are given an upper time bound on 1800 seconds. First we analyze the exact solution methods, i.e., we apply `CPLEX` on the mathematical formulation (1)-(9) and compare with the branch-and-cut-and-price algorithm. Then the heuristics are considered.

CPLEX
JSCNR can be solved to optimality by generating the edge based model (1)-(9) for each instance and then using `CPLEX` to solve the model. Test results are seen in Table 1. The results show that `CPLEX` runs out of memory or time even for the smaller instances. This motivates the need for a more sophisticated exact solution method.

Exact
Solving JSCNR with `CPLEX` was unsuccessful; hence we implemented the more sophisticated branch-and-cut-and-price (BCP) algorithm from Section 4. Test results are seen in Table 2.

The results show that the sophisticated BCP algorithm is also unable to solve several instances within the 1800 seconds. It does, though, generally perform better than when using `CPLEX`, both with respect to time usage and to the number of solved instances. An in-depth analysis of the test results for the BCP algorithm has shown that the bottleneck is solving the pricing problem to

Jobs	Res.	BW	Results $\lambda = 5$	Time	Results $\lambda = 10$	Time	Results $\lambda = 20$	Time
10	10	10	12	0.18	12	0.30	12	0.65
10	10	25	12	0.17	12	0.33	12	0.66
10	20	10	2	0.85	2	1.77	2	3.57
10	20	25	2	0.88	2	1.70	2	3.50
10	50	10	69	31.14	69	65.98	69	155.35
10	50	25	69	31.29	69	66.19	69	157.83
10	100	10	7	177.31	–	oom	–	oom
10	100	25	7	179.32	–	oom	–	oom
20	10	10	26	1.12	26	0.94	26	1.95
20	10	25	26	0.50	26	0.92	26	1.91
20	20	10	63	4.65	63	9.09	63	18.40
20	20	25	63	4.61	63	8.93	63	18.30
20	50	10	159	108.75	–	oom	–	oom
20	50	25	159	108.10	–	oom	–	oom
50	10	10	80	1.20	80	1.96	80	3.80
50	10	25	80	1.09	80	1.89	80	3.67
50	20	10	153	6.50	153	17.85	153	36.28
50	20	25	153	9.64	153	17.90	153	37.98
50	50	10	–	oom	–	oom	–	oom
50	50	25	–	oom	–	oom	–	oom
100	10	10	147	6.94	147	8.99	147	14.84
100	10	25	147	4.94	147	8.26	147	15.29
100	20	10	285	36.85	285	139.70	285	151.07
100	20	25	285	40.36	285	64.20	285	150.36
100	50	10	–	oom	–	oom	–	oom
100	50	25	–	oom	–	oom	–	oom
200	10	10	164	7.66	164	8.86	164	14.77
200	10	25	164	6.54	164	8.77	164	15.14
200	20	10	316	71.97	316	122.54	316	223.24
200	20	25	316	82.61	316	116.43	316	218.04
200	50	10	–	oom	–	oom	–	oom
200	50	25	–	oom	–	oom	–	oom
10	14	10	41	0.67	41	1.34	41	2.53
10	14	25	41	0.66	41	1.28	41	2.53
20	14	10	116	1.57	116	3.47	116	5.24
20	14	25	116	1.54	116	3.35	116	5.39
50	14	10	266*	1866.91*	272*	1815.78*	273*	1922.77*
50	14	25	266*	1810.28*	272*	1810.80*	273*	1808.40*

Table 1: Test results for the CPLEX approach. The first three columns hold information on the number of jobs, resources and the amount of bandwidth. Instances with 14 resources are of type NDGF; all other instances are of the Tandem type. Then follows two columns for three different wavelength settings, i.e., number of wavelengths per fiber: $\lambda = 5, 10$, and 20. The two columns for each setting give the result value and the running time in seconds. An entry with 'oom' means that the instance could not be solved due to memory problems (Out Of Memory). An entry with '*' indicates that the instance could not be solved within 1800 seconds and thus ran out of time. The best feasible solution is then given.

Jobs	Res.	BW	Results $\lambda = 5$	Time	Results $\lambda = 10$	Time	Results $\lambda = 20$	Time
10	10	10	12.00	0.01	12.00	0.00	12.00	0.00
10	10	25	12.00	0.00	12.00	0.00	12.00	0.00
10	20	10	2.00	0.00	2.00	0.01	2.00	0.00
10	20	25	2.00	0.00	2.00	0.00	2.00	0.00
10	50	10	69.00	0.02	69.00	0.01	69.00	0.02
10	50	25	69.00	0.01	69.00	0.02	69.00	0.02
10	100	10	7.00	0.04	7.00	0.05	7.00	0.05
10	100	25	7.00	0.05	7.00	0.06	7.00	0.05
10	500	10	4.00	5.27	4.00	5.39	4.00	5.44
10	500	25	4.00	5.26	4.00	5.37	4.00	5.46
20	10	10	26.00	0.02	26.00	0.05	26.00	0.09
20	10	25	26.00	0.02	26.00	0.04	26.00	0.09
20	20	10	63.00	0.05	63.00	0.11	63.00	0.23
20	20	25	63.00	0.06	63.00	0.11	63.00	0.23
20	50	10	159.00	5.36	159.00	15.08	159.00	48.43
20	50	25	159.00	5.35	159.00	15.20	159.00	48.18
20	100	10	134.00	4.84	134.00	10.98	134.00	27.68
20	100	25	134.00	4.85	134.00	11.01	134.00	27.58
20	200	10	39.00	0.41	39.00	0.41	39.00	0.40
20	200	25	39.00	0.40	39.00	0.37	39.00	0.41
50	10	10	80.00	0.00	80.00	0.01	80.00	0.01
50	10	25	80.00	0.00	80.00	0.00	80.00	0.01
50	20	10	134.00	1800.28*	148.00	1800.71*	148.00	1800.39*
50	20	25	134.00	1800.18*	148.00	1800.05*	148.00	1800.91*
50	50	10	275.00	56.49	275.00	16.98	275.00	48.96
50	50	25	298.00	26.83	314.00	9.65	314.00	26.71
50	100	10	166.00	5.35	166.00	11.92	166.00	29.77
50	100	25	166.00	5.36	166.00	12.07	166.00	29.93
50	200	10	69.00	4.34	69.00	8.22	69.00	16.38
50	200	25	69.00	4.31	69.00	8.17	69.00	16.25
100	10	10	147.00	0.98	147.00	1.91	147.00	4.27
100	10	25	147.00	0.98	147.00	1.94	147.00	4.25
100	20	10	285.00	4.05	285.00	9.85	285.00	27.62
100	20	25	285.00	4.09	285.00	10.00	285.00	27.53
100	50	10	713.00	1809.70*	738.00	1835.00*	738.00	2001.58*
100	50	25	801.00	1802.84*	807.00	1605.63	738.00	2006.66*
100	100	10	810.00	1800.29*	685.00	1800.39*	749.00	1815.39*
100	100	25	619.00	1800.93*	743.00	1803.58*	794.00	1803.44*
100	200	10	240.00	1801.48*	240.00	1820.37*	240.00	1802.53*
100	200	25	240.00	1811.03*	240.00	1808.99*	240.00	1811.26*
100	500	10	219.00	6.00	219.00	5.96	219.00	5.33
100	500	25	219.00	6.02	219.00	6.04	219.00	5.90
200	10	10	148.00	1800.43*	148.00	1800.51*	148.00	1801.23*
200	10	25	148.00	1800.15*	148.00	1800.04*	148.00	1800.92*
200	20	10	296.00	1802.13*	316.00	1800.37*	316.00	1802.04*
200	20	25	296.00	1800.97*	316.00	1801.71*	316.00	1800.90*
200	50	10	347.00	1806.79*	499.00	2134.76*	626.00	5410.29*
200	50	25	370.00	1915.16*	535.00	1921.21*	669.00	4224.28*
200	100	10	354.00	1817.09*	528.00	1836.88*	480.00	2077.20*
200	100	25	354.00	1875.56*	486.00	1860.29*	535.00	1912.69*
200	200	10	371.00	193.02	371.00	36.72	371.00	80.67
200	200	25	371.00	17.86	371.00	36.85	371.00	80.45
200	500	10	227.00	60.40	234.00	123.38	234.00	251.89
200	500	25	227.00	60.48	234.00	123.26	234.00	251.50
10	14	10	41.00	0.00	41.00	0.00	41.00	0.00
10	14	25	41.00	0.00	41.00	0.00	41.00	0.00
20	14	10	116.00	0.01	116.00	0.00	116.00	0.00
20	14	25	116.00	0.00	116.00	0.00	116.00	0.00
50	14	10	295.00	0.02	295.00	0.01	295.00	0.02
50	14	25	295.00	0.02	295.00	0.01	295.00	0.02
100	14	10	555.00	8.90	555.00	14.90	555.00	28.41
100	14	25	555.00	8.56	555.00	14.97	555.00	28.04
200	14	10	668.00	2677.35*	658.00	3736.75*	658.00	3053.28*
200	14	25	668.00	2010.45*	658.00	3777.48*	658.00	3033.20*

Table 2: Results for the exact solution approach on the tandem instances (top) and the NDGF instances (bottom). The first two columns hold the number of jobs and resources for the instance, respectively. Instances with 14 resources are of type NDGF, all other instances are of the Tandem type. The third column gives information on the amount of bandwidth per edge. Then follows two columns for three different wavelength settings, i.e., number of wavelengths per fiber: $\lambda = 5, 10,$ and 20. The two columns for each setting give the result value and the running time in seconds. When time usage finishes with a star ('*'), then the test has run out of time.

optimality. Recall that the pricing problem is the RWA over time, which is $\mathcal{NP}$-hard. The BCP algorithm solves the pricing problem heuristically until no columns are found at which point the pricing problem is solved to optimality. Separating cuts, solving the master problem, generating branching candidates, and branching take little time and the search tree is always small.

When comparing results for the tandem instances with results for the NDGF instances, we see that the BCP algorithm has equal difficulty with solving both instance types. The topology of the NDGF instances can be viewed as a combination of a tree and a star structure and not many paths share edges. Conversely, paths share many edges in the tandem instances. The reason why the BCP algorithm finds both instances hard to solve is probably that both the scheduling and the routing problem are $\mathcal{NP}$-hard, hence if either constitutes a bottleneck then the overall problem is very difficult to solve.

Heuristics

Solving JSCNR to optimality is very difficult even for smaller instances. Hence heuristics for the problem may be useful when larger instances are to be solved. The proposed heuristics in Section 3 have been implemented. First they are compared with the exact solution approach and then they are compared with each other. See the tables at `http://www.diku.dk/~gamst/tables.pdf` for detailed test results.

An overview of comparing the heuristics with the branch-and-cut-and-price algorithm can be seen in Table 3. The table illustrates average solution value gaps and time usages for instances, which the exact algorithm has solved to optimality. As can be seen in the table, the heuristics only use a very small fraction of time compared to the exact approach. The solution value gap is never larger than 16%. For the grid heuristics, First Come First Serve has best performance, followed by Random Fit, Best Fit and First Fit. Fixed-Alternate Routing with 2 paths per data connection finds the smallest gaps, followed by Fixed-Alternate with 5 paths per connection, 1 path per connection and finally Adaptive Routing. No clear pattern emerges when considering wavelength assignment. For the First Come First Serve and Random Fit grid heuristics, First Fit wavelength assignment performs well. Otherwise Most Used has good performance.

		R=FA, p=1		R=FA, p=2		R=FA, p=5		R=A	
Grid	WA	Solution	Time	Solution	Time	Solution	Time	Solution	Time
FCFS	FF	3.46%	<0.01%	2.89%	<0.01%	3.17%	<0.01%	6.63%	<0.01%
FCFS	MU	3.46%	<0.01%	4.12%	<0.01%	4.40%	<0.01%	7.86%	<0.01%
FCFS	RF	4.95%	<0.01%	3.01%	<0.01%	3.29%	<0.01%	7.13%	<0.01%
BF	FF	10.06%	<0.01%	9.48%	<0.01%	10.05%	<0.01%	13.74%	<0.01%
BF	MU	8.95%	<0.01%	8.37%	<0.01%	8.94%	<0.01%	12.63%	<0.01%
BF	RF	9.61%	<0.01%	8.37%	<0.01%	8.94%	<0.01%	13.01%	<0.01%
FF	FF	11.65%	<0.01%	10.80%	<0.01%	10.04%	<0.01%	15.32%	<0.01%
FF	MU	10.54%	<0.01%	9.69%	<0.01%	8.93%	<0.01%	14.21%	<0.01%
FF	RF	11.19%	<0.01%	9.69%	<0.01%	8.93%	<0.01%	14.59%	<0.01%
RF	FF	4.11%	<0.01%	3.80%	<0.01%	4.52%	<0.01%	9.58%	<0.01%
RF	MU	3.46%	<0.01%	5.60%	<0.01%	5.49%	<0.01%	13.22%	<0.01%
RF	RF	6.25%	<0.01%	4.13%	<0.01%	5.03%	<0.01%	10.55%	<0.01%

Table 3: Performance of the heuristics compared to the exact results. The first two columns denote the grid and the wavelength assignment heuristics, respectively. Then follows pairs of comparison results, where the difference is measured in percent: the first column holds the average gap between the optimal and heuristic solution values and the second column holds the average percentage of the exact solution time used by the heuristic. `R` stands for routing, and the options are `FA` (Fixed-Alternate) and `A` (Adaptive). `p` denotes the number of paths generated per data connection.

The Table only reports average gaps for some instances; hence it does not give a full picture of the performance of the heuristics. This is determined next when comparing the heuristics to each other. An overview of this comparison is seen in Table 4. The summary is based on ranking the performance of the heuristics: the lower the rank the better performance. The average ranking of solution values for all instances is given in the `Solution` columns of the table and the average ranking of running times is given in the `Time` columns. An overview of actual time usage is seen in Figure 3-5.

Grid	WA	R=FA, p=1 Solution	R=FA, p=1 Time	R=FA, p=2 Solution	R=FA, p=2 Time	R=FA, p=5 Solution	R=FA, p=5 Time	R=A Solution	R=A Time
FCFS	FF	0.86	2.19	0.70	2.15	0.81	2.41	0.90	2.38
FCFS	MU	0.99	3.06	0.86	3.23	0.92	2.88	0.94	3.75
FCFS	RF	0.99	1.64	0.77	1.68	0.81	2.61	0.97	2.32
BF	FF	1.97	3.45	1.79	3.53	2.09	3.83	1.37	3.52
BF	MU	1.88	4.27	1.73	4.27	2.03	3.86	1.40	4.61
BF	RF	1.92	2.83	1.73	3.08	2.06	3.91	1.43	3.57
FF	FF	2.59	3.54	2.35	3.89	2.20	3.95	2.11	3.73
FF	MU	2.48	4.30	2.34	4.36	2.18	4.01	2.13	4.71
FF	RF	2.51	3.09	2.29	3.17	2.18	4.19	2.13	3.72
RF	FF	1.75	2.01	1.64	1.97	1.65	2.24	1.77	2.11
RF	MU	1.87	3.17	1.80	3.33	1.83	3.04	1.89	3.78
RF	RF	1.96	1.38	1.76	1.40	1.74	2.35	1.90	2.10
FCFS	FF	3.29	1.69	3.28	1.70	3.58	1.86	3.28	1.69
FCFS	MU	3.29	2.79	3.28	3.17	3.58	3.35	3.28	2.76
FCFS	RF	3.29	1.79	3.28	1.94	3.58	2.06	3.28	1.62
BF	FF	0.56	2.72	0.56	3.20	0.56	3.85	0.58	3.53
BF	MU	0.56	5.02	0.56	5.00	0.56	5.36	0.58	4.99
BF	RF	0.56	3.47	0.56	3.58	0.56	3.93	0.58	3.68
FF	FF	2.66	4.07	2.06	4.47	2.23	4.22	2.91	4.05
FF	MU	2.66	5.44	2.06	5.44	2.23	6.34	2.91	5.12
FF	RF	2.66	4.65	2.06	4.08	2.23	4.69	2.91	3.88
RF	FF	1.18	1.19	1.30	1.70	1.46	1.72	1.32	1.57
RF	MU	1.45	3.59	1.15	2.94	1.19	3.48	1.20	3.11
RF	RF	1.15	1.56	1.43	2.05	1.27	1.94	1.50	1.92

Table 4: Performance of the heuristics having been ranked for best time and solution value for the tandem (top) and the NDGF (bottom) instances. The table displays the average ranking, `R` stands for routing, and the options are `FA` (fixed-alternate) and `A` (adaptive). `p` denotes the number of paths generated per data connection.

The ranked results and the time usage illustrations are analyzed with respect to each of the main three heuristic approaches: the overall grid heuristic, the routing heuristic and the wavelength assignment heuristic.

For the NDGF instances we see that the Best Fit grid heuristic gives better solution values than Random Fit, followed by the First Fit and First Come First Serve heuristics. The wavelength assignment heuristics perform equally well. For the routing strategy the best setting seems to be using Fixed-Alternate routing with 2 paths per data connection. Looking at time usage, then the First Come First Serve and Random Fit grid heuristics perform better than both Best Fit and First Fit. However, the graph in Figure 3 illustrates that the time difference is small. The Most Used assignment generally requires more time than the other two wavelength assignment strategies, but again the time difference is small as seen in Figure 4. Finally, Adaptive routing uses less time than Fixed-Alternate, which becomes more time consuming as the number of generated paths per data connection increases. The time difference in small, see Figure 5.

For the tandem instances the First Come First Serve grid heuristic finds the best solution values. All wavelength assignment strategies seem to perform equally well with respect to solution values, while the Adaptive and Fixed-Alternate routing with 2 paths finds better solutions than other strategies. Looking at time usage, the First Come First Serve and Random Fit strategies are the faster grid heuristics, especially for the large instances with 500 jobs. Most Used requires more time than the other wavelength assignment strategies and for the large tandem instance, the time difference is significant. The Fixed-Alternate becomes more time consuming as the number of generated paths per data connection increases and the Adaptive routing is even slightly slower. The time difference between routing heuristics is insignificant, though, which can be seen in Figure 5.

Looking at general time usage, the practical running times reflect the theoretical running times from Section 3.1. For wavelength assignment this means that the Most Used strategy generally

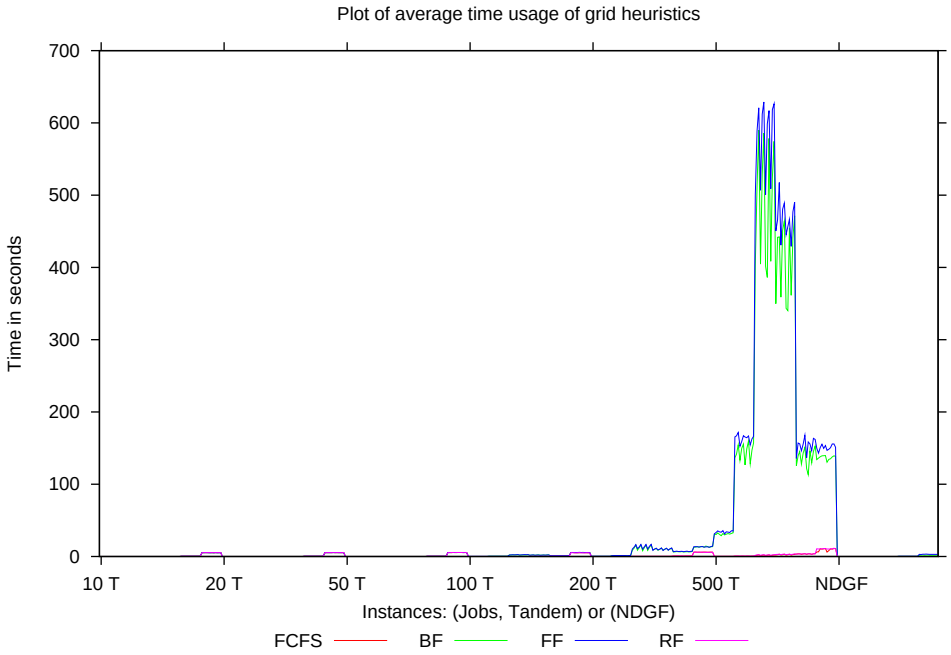


Figure 3: Illustration of time usage in seconds for the grid heuristics. The x-axis denotes instances, where the first number indicates the number of jobs for the tandem instances and where the last part indicates the NDGF instances. Plots for instances with the denoted number of jobs and with 10-500 resources are given between two tics on the x-axis.

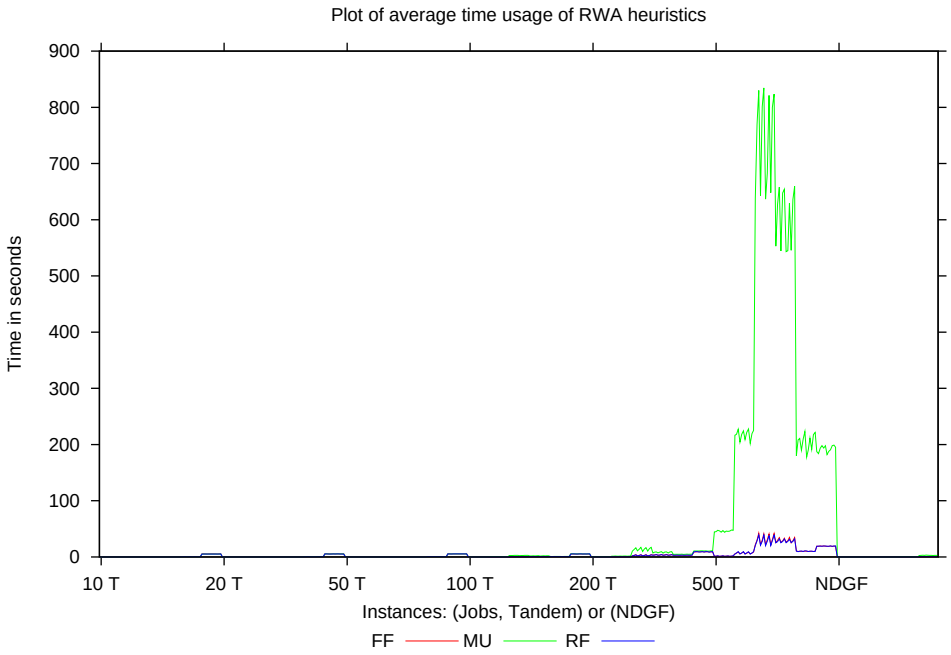


Figure 4: Illustration of time usage in seconds for the RWA heuristics. The x-axis denotes instances, where the first number indicates the number of jobs for the tandem instances and where the last part indicates the NDGF instances. Plots for instances with the denoted number of jobs and with 10-500 resources are given between two tics on the x-axis.

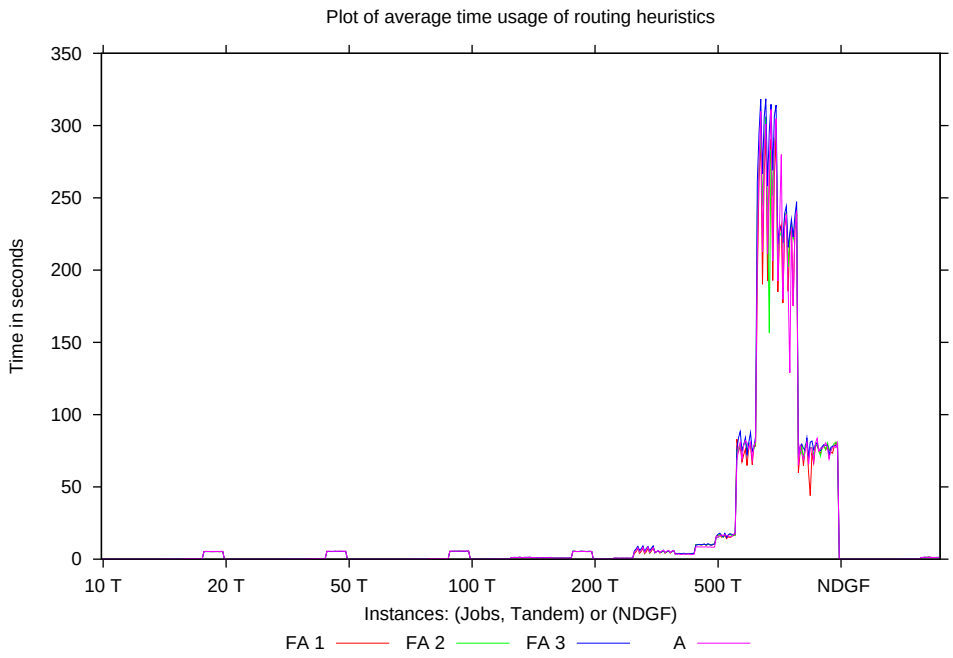


Figure 5: Illustration of time usage in seconds for the routing heuristics. The x-axis denotes instances, where the first number indicates the number of jobs for the tandem instances and where the last part indicates the NDGF instances. Plots for instances with the denoted number of jobs and with 10-500 resources are given between two tics on the x-axis.

requires more time than First Fit and Random Fit. Adaptive routing is generally faster than Fixed-Alternate routing whose running time increases with the number of generated paths per data connection. Finally, the First Come First Serve and Random Fit grid heuristics have smaller time usage than Best Fit and First Fit.

Comparing the heuristics with each other gives a slightly different pattern than when comparing heuristics with the exact solution results. This is due to two reasons. 1: Not all instances were considered when comparing with exact results, because the BCP algorithm was not able to solve all instances. 2: The average gap may be large if a heuristic gives very poor results for few instances but good results for all other instances. The ranking system does not care how far off a result may be and does thus not punish very poor performance equally hard.

Overall analysis

Using a black-box strategy for solving JSCNR may not always be the best choice. Instead the grid administrator should identify the current bottlenecks with respect to scheduling and network usage in order to find a good heuristic. The Best Fit grid heuristic utilizes available resources well for instances with no or little network problems. This is concluded from considering the NDGF instances, where paths share few edges. A reason for this is that Best Fit makes sure that jobs are placed according to them taking up as little time space in the network and on the resource as possible, hence giving good resource utilization. When the network constitutes a significant bottleneck, then First Come First Serve makes sure that jobs are forwarded to execution soon after arrival, which yields the best solution values. This is seen in the Tandem instances, where paths share many edges. A reason for this is that the strategy uses network wavelengths as early as possible instead of at some later time; when the latter is the case, then the time slots at which wavelengths become available after a subset of jobs are assigned, may become so small that data for the remaining jobs cannot arrive at the executing resource in time. Time usage must also be taken into account. If the grid system consists of many resources and/or many jobs, then it may

be beneficial to choose a more straightforward grid heuristic like First Come First Serve, regardless of network constraints.

Most Used wavelength assignment may often give better results than both First Fit and Random Fit but also requires more time. The reason for the better results is that by choosing the most used wavelength, more wavelengths may be available for the next data connection request. First Fit and Random Fit assignment generally perform equally well both with respect to solution values and time usage.

Generally, the best routing strategies with respect to solution values are Fixed-Alternate routing with 2 paths per data connection closely followed by Adaptive Routing and Fixed-Alternate with 5 paths per data connection. The Fixed-Alternate routing considers previously routed data connection and thus has good performance when generating more than 1 path per data connection.

A final recommendation is based on the comparison with exact solution values in Table 3, on the comparison of heuristics in Table 4 and on time usage in Figure 3-5. We suggest using First Come First Serve grid scheduling, Fixed-Alternate routing with 2 paths per data connection and First Fit wavelength assignment. This setting generally gives lower gaps compared to exact values and it also generally has best performance when only considering the heuristics. The solution approach, however, should be decided based on an analysis of the grid topology and the expected CPU and network load. For small instances, an exact solution can be found within reasonable time. For larger instances some consideration should be given on which grid heuristic is more appropriate.

6 Conclusion

This paper introduced The Integrated Job Scheduling and Constrained Network Routing Problem (JSCNR) with application in production planning and telecommunication. JSCNR was formally presented and a mathematical formulation was given. JSCNR was shown to be $\mathcal{NP}$-hard, as it holds both the $\mathcal{NP}$-hard Integrated Job Scheduling and Network Routing Problem (JSNR) and the $\mathcal{NP}$-hard Routing and Wavelength Assignment Problem (RWA) as special cases.

In this paper we formalized JSCNR and presented a mathematical formulation. A branch-and-cut-and-price (BCP) algorithm was presented, where the pricing problem assigns a job to a machine and the master problem merges the job assignments into an overall feasible solution. Finally, a number of heuristics for JSNR was presented along with a number of heuristics for RWA and they were merged into a total of 24 different solution methods for JSCNR.

The proposed methods were computationally evaluated on two types of test instances: a tandem topology with 10-500 machines and a real-life network topology taken from the Nordic DataGrid Facility with 14 machines.

Using `CPLEX` to solve the mathematical formulation yielded somewhat poor results as only smaller instances were solved due to memory and time problems. The BCP algorithm was capable of solving more instances, however, it still timed out for several instances because its pricing problem is $\mathcal{NP}$-hard.

All heuristics were tested and compared with the exact approach and with each other. The computational results showed that First Come First Serve job assignment heuristic gave best results along with the routing strategy, which proposes two routes for each demand. The running times of the computational evaluations reflected the theoretical running times for the heuristics well. Furthermore, all instances were solved within minutes.

Future work on JSCNR could concentrate on finding optimal solutions. The proposed decomposition resulted in an $\mathcal{NP}$-hard pricing problem, which caused time issues. Future work could consider other decompositions with possibly less complex pricing problems.

It would also be relevant to consider metaheuristics, e.g., local search methods. The heuristics presented in this work could be used as base in metaheuristics. It is expected that metaheuristics

would improve the solution quality, but would also have greater running times. Metaheuristics are expected to provide a good alternative with performance lying between that of the greedy heuristics and of the BCP algorithm with respect to solution quality and time usage.

Acknowledgement
We would like to thank GlobalConnect A/S for their support of this work.

References

[1] V. Agarwal, G. Dasgupta, K. Dasgupta, A. Purohit, and B. Viswanathan. Deco: Data replication and execution co-scheduling for utility grids. In *ICSOC 2006, LNCS 4294*, pages 52–65, 2006.

[2] N. Banerjee, V. Metha, and S. Pandey. A genetic algorithm approach for solving the routing and wavelength assignment problem in wdm network. In *3rd IEEE/IEE International Conference on Networking, ICN 2004, Paris*, pages 70–78, 2004.

[3] R. J. Bates. *Optical Switching and Networking Handbook*. McGraw-Hill Professional Publishing, 2001.

[4] A. Birman and A. Kershenbaum. Routing and wavelength assignment methods in single-hop all-optical networks with blocking. In *IEEE Infocom'95*, 1995.

[5] CERN. Worldwide LHC computing grid. `http://lcg.web.cern.ch/LCG/`.

[6] I. Chlamtac, A. Ganz, and G. Karmi. Lightpath communications: an approach to high bandwidth opticalwan's. *IEEE Transactions on Communications*, 40(7):1171–1182, 1992.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. The MIT Press and McGraw-Hill, second edition, 2001.

[8] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Math*, 194:229 – 237, 1997.

[9] A. Elghirani, R. Subrata, and A. Y. Zomaya. Intelligent scheduling and replication in datagrids: a synergistic approach. In *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, 2007.

[10] D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998.

[11] M. Gamst. Heuristics for the offline scheduling problem in the minimum intrusion grid. In Submission, December 2008.

[12] M. Gamst and D. Pisinger. Integrated job scheduling and network routing. In submission, September 2009.

[13] M. Grønager. A nordic tier-1 for LHC. Conference Presentation, GLORIAD, 2007.

[14] J. Iness and B. Mukherjee. Sparse wavelength conversion in wavelength-routed wdm optical networks. *Photonic Network Communications*, 1(3):183–205, 1999.

[15] M. Kovacevic and A. Acampora. Benefits of wavelength translation in all-optical clear-channel networks. *IEEE Journal on Selected Areas in Communications*, 14:868–880, 1996.

[16] L. Marchal, Y. Robert, P. V.-B. Primet, and J. Zeng. Scheduling network requests with transmission window. Technical Report 2005-32, Laboratoire de L'Informatique du Parallélisme, 2005. July.

[17] B. Ramamurthy and B. Mukherjee. Wavelength conversion in wdm networking. *IEEE Journal on selected areas in Communications*, 16:1061–1073, 1998.

[18] L.-M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. Technical Report Publication CRT-2003-39, Centre de recherche sur les transports, Université de Montréal, 2003.

[19] R. B. Sørensen. Analysing resource allocation in minimum intrusion grid (mig) using mechanism design. Master's thesis, Department of Computer Science, Copenhagen University (DIKU), 2007.

[20] S. Subrarnaniam and R. A. Barry. Wavelength assignment in fixed routing wdm networks. In *IEEE Int. Conf. Communications*, pages 406–410, 1997.

[21] E. Varvarigos, V. Sourlas, and K. Christodoulopoulos. Routing and scheduling connections in networks that support advance reservations. *Computer Networks*, 52:2988–3006, 2008.

[22] H. Zang, J. P. Jue, and B. Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical wdm networks. *Optical Networks Magazine*, 1:47–60, January 2000.

This report examines the NP-hard problem of scheduling a number of jobs on a finite set of machines such that the overall profit of executed jobs is maximized. Each job demands a number of resources, which must be sent to the executing machine via constrained paths. Furthermore, two resource demand transmissions cannot use the same edge in the same time period.
An exact solution approach based on Dantzig-Wolfe decomposition is proposed along with several heuristics. The methods are computationally evaluated on test instances arising from telecommunications with up to 500 jobs and 500 machines. Results show that solving the problem to optimality is very difficult. The proposed heuristics have good performance with an average solution value gap of 3 % and with very small running times.

ISBN 978-87-90855-67-3

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

Produktionstorvet
Building 424
DK-2800 Kongens Lyngby
Denmark
Tel. +45 45 25 48 00
Fax +45 45 93 34 35

www.man.dtu.dk