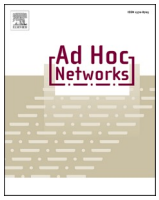
Ad Hoc Networks 151 (2023) 103289

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc



A delay-tolerant network approach to satellite pickup and delivery scheduling

Christopher John Lowe [a], Ruaridh Alexander Clark [a], Ciara Norah McGrath [b], Malcolm Macdonald [a,*]

[a] Applied Space Technology Laboratory (ApSTL), Department of Electronic and Electrical Engineering, University of Strathclyde, 204 George St, Glasgow, G1 1XW, United Kingdom

[b] Space Systems Research Group, School of Engineering, University of Manchester, Oxford Rd, Manchester, M13 9PL, United Kingdom

ARTICLE INFO

Keywords:
Scheduling
Tasking
delay-tolerant networks
Satellite networks
Contact Graph Routing

ABSTRACT

An approach to job scheduling and task allocation in delay- and disruption-tolerant satellite networks, called Contact Graph Scheduling (CGS) is introduced. This method aims to minimize latency between the arrival of requests for satellite data products, and the delivery of those products, identifying efficient pickup and delivery paths through a resource- constrained time-varying network. Requests for goods are processed as soon as they arrive, resulting in the discovery of task distribution and product delivery routes, while being aware of both resource and operational constraints. CGS sits at the interface between Contact Graph Routing (CGR), the dynamic pickup and delivery problem (dPDP) and satellite scheduling. CGS provides a combined scheduling-routing solution, such that the route to be taken by data toward their destination informs the allocation of pickup tasks to network nodes. Results show that CGS can increase the number of fulfilled customer requests, while maintaining acceptable levels of delivery latency, compared to other, less burdensome methods. This study highlights the impact of contact reliability on CGS performance, suggesting that as uncertainty increases, so does the benefit of employing locally, rather than centrally, defined data routing strategies.

1. Introduction

Tasking of satellites to acquire data has been an active field of operations research and combinatorics for many years. The objective is typically to assign data acquisition tasks to satellite nodes in a manner that maximizes (or minimizes) some value metric across the whole mission. The exact nature of this value metric is application-dependent, but is often closely related to data delivery, either throughput or latency, or task-specific profit.

In this work, an approach to satellite scheduling, called Contact Graph Scheduling (CGS), is introduced that provides task allocation in response to the submission of pickup and delivery requests, for systems that exhibit delay- and disruption-tolerant network (DTN) characteristics. This approach schedules requests immediately, upon their arrival, considering the full product life-cycle including; request, task distribution, data pick-up and data delivery. Scheduling in this manner, using a DTN shortest-path inspired approach, offers an alternative, more compute-efficient solution compared to traditional, linear programming-based methods for satellite scheduling, in which requests are grouped and solved for some global optimality. This method, which integrates both task scheduling and data routing at its core, is optimal at the task-level and could be deployed in a decentralized manner without significant overhead on resource-constrained nodes. Furthermore, the impact of handling requests immediately, rather than as a group, enables the provision of a service level guarantee to the request provider, which could offer benefits in certain situations, including commercial, scientific or defence applications.

CGS offers the first known method of task scheduling in satellite networks via a delay-tolerant network data routing approach. This approach drastically reduces computational overhead in comparison to more traditional, linear programming approaches to satellite task scheduling, offering a viable solution for large scale networks with intermittent connectivity between space and ground nodes.

* Corresponding author
E-mail address: malcolm.macdonald.102@strath.ac.uk (M. Macdonald).

https://doi.org/10.1016/j.adhoc.2023.103289
Received 13 April 2023; Received in revised form 3 August 2023; Accepted 2 September 2023
Available online 6 September 2023

1.1. Task Scheduling in Satellite Networks

The typical lifecycle of data products in satellite networks is: commands (i.e. tasks) are uploaded to the satellite from a ground-based gateway; data is acquired/generated by the on-board payload when the satellite makes contact with the target; and, data is downloaded to a gateway for further processing, before reaching the customer. Inter-satellite ("cross") links (ISLs), if available, provide additional data-relay opportunities that could expedite the transfer of data following its generation, reducing overall latency between pickup and delivery.

Until the 1980s, the challenge of satellite scheduling was addressed by human-in-the-loop approaches [8]. Early work aimed at automating the scheduling process formulated an Integer Linear Programming (ILP) problem as a *single machine scheduling model* [22]. They derived a set of heuristics and dynamic programming solutions to overcome the computation burden of solving the linear program, which were addressed shortly after using ILP methods in Bensana et al. [5]. More recently, the allocation of data pickup tasks has been achieved through some form of exact or inexact optimization, either Mixed Integer Linear Programming (MILP) [3,36], Dynamic Programming (DP) [31] or some combination of the two [10]. Justification for using MILP approaches is the desire to reach a globally optimal solution for the allocation of a set of tasks that are known prior to completing the scheduling procedure. An alternative approach to generating a schedule for satellite operations makes use of the *unrelated parallel machine scheduling with time windows* problem [43]. This approach incorporates a generic MILP formulation to maximize total mission profit, with problem space reduction achieved via a pre-processing phase to ease computational demand. Optimal solutions for up to 800 tasks on 20 satellites has been demonstrated in Wang et al. [43], but with no consideration of inter-satellite communication, on-board resource constraints, or distribution of task information (i.e. an assumption is made that tasks can reach the satellites prior to data acquisition).

Graph-based heuristic approaches have also been implemented to solve the satellite scheduling problem, and the use of *maximum independent sets* offers an interesting alternative to traditional numerical optimization [17]. The approach models data acquisition opportunities as nodes in a network, with adjacent nodes (i.e. pairs that are connected by edges) representing acquisition events that cannot *both* be completed. This reduces the problem to one of finding the maximum independent set in the graph, that is, finding the set of non-adjacent nodes that exhibit the maximum node-count. This results in finding the maximum number of tasks that can be feasibly carried out by the system, limiting the interpretation of mission "value" to a function of task completion count.

Consideration of the data download activity is often overlooked in the satellite scheduling literature, such that pickup of data is considered the end of the process and is the sole contributor to the mission's objective. If data latency is important, then the duration between acquisition and delivery of the acquired data must be included in the analysis. In some cases, download times are incorporated as a means to ensure delivery of the data is feasible [24] and in others it forms a key part of the value proposition [10]. While delivery has been considered in these cited examples, inclusion of how tasks physically reach the satellites is not. Rather, an assumption is made that tasks can be uploaded prior to the pickup operations, likely during some prior contact with a gateway. This is not unreasonable in the sense that one could simply wait until the next gateway contact before executing the optimization, however in the dynamic scheduling case, where requests are to be processed immediately upon arrival, this is insufficient. Furthermore, consideration of task distribution through multi-hop (ISL-enabled) paths has not been incorporated in a solution to the satellite scheduling problem. This extension to traditional direct upload is a key consideration for operators of satellite constellations with inter-satellite communication capabilities that aim to offer data services with a dependency on delivery latency, and is included in this work.

1.2. The Pickup and Delivery Problem

The *Pickup and Delivery Problem* (PDP), a sub-set of the more general *Vehicle Routing Problem* (VRP) family, aims to minimize distance travelled (or time taken) for one or more vehicles to carry out pickup of goods from their respective depots and delivery of those goods to their respective destinations [37]. This has clear similarities to the satellite scheduling problem being addressed in this work, which considers the pickup of data by satellites from targets, and delivery of that data by satellites to gateways. Indeed, mobility patterns in the satellite case are restricted, that is one cannot simply direct the satellite into a vastly different orbit at will, such that the problem is constrained. Focus, in the satellite case, is therefore more on the allocation of pickup and delivery activities that fit with the satellite trajectory plan.

Extensions to the original PDP provide further correlations, namely the consideration of goods transfer between vehicles, the PDP with transshipments (PDP-T), and the consideration of activity-specific time-windows, the PDP with time windows (PDPTW). Methods addressing the *Pickup and Delivery Problem with Time-Windows and Transshipments* (PDPTWT) have evolved over the years, from an iterative, 2-phase heuristic method that provides sub-optimal solutions [30], to mixed integer linear programming formulations that consider a broad set of system constraints and objectives [28,34,44]. The current state of the art, discussed in Lyu and Yu [28], demonstrates optimal solutions for scenarios featuring up to 25 requests and 2 transshipment stations. The satellite constellation applications being addressed here, are orders of magnitude greater in terms of both transshipment opportunities and request count.

A more relevant sub-class of VRPs is the *dynamic pickup and delivery problem* (dPDP) [6], in which requests for goods arrive throughout the operational period and must be processed as they arrive. Such terrestrial applications include inner-city courier services and the "dial-a-ride problem" (DARP) associated with taxi allocations [15,33]. As with traditional PDP applications, the goal is to identify vehicle routes that maximize some objective function, which typically presents a large, complex problem space that scales poorly with network size. The dPDP with transfers (dPDPT) is considered in Bouros et al. [11] and Andini et al. [2] and with time windows (dPDPTW) in Karami et al. [23]. In Bouros et al. [11], handling incoming requests that render the original pickup and delivery plan out of date is treated as a shortest path problem. However, they show that their conceptual graph, describing the paths over which vehicles can travel, does not exhibit the *principal of optimality* necessary for use of traditional shortestpath algorithms, such as Dijkstra [16] or Bellman-Ford [4]. This is not the case in the satellite-specific dPDP, with the deterministic, largely non-changing relative mobility of satellite and gateway nodes offering a key difference that enables paths on a time-varying graph to be *identified* using these traditional methods. This approach is presented here, offering a solution to an adapted version of the *Dynamic Pickup and Delivery Problem with Time Windows and Transshipments* (dPDPTWT).

1.3. Delay-Tolerant Network Routing

A network that exhibits intermittent and/or disrupted connectivity, generally with few end-to-end paths between node-pairs at any particular moment in time, can be referred to as a "challenged" network. These types of networks present environments in which the use of traditional protocols, such as TCP/IP, is difficult or indeed impossible. To overcome this challenge, use of the Delay-tolerant Network Architecture [39] and Bundle Protocol [12], enables data transfer through such a network, resulting in the formation a delay-tolerant network (DTN). DTNs are unlike internet architectures that can be described as connected graphs, where two-way communication between any node-pair is generally possible, such that attributes like link-state establishment and send-receive acknowledgement are expected. Challenged networks are common in every- day life, with terrestrial

examples including human movement through public transport systems and message forwarding through social interactions, and in non-terrestrial examples including the dissemination of data through deep-space and low Earth orbit (LEO) satellite networks. In the context of this work, the networks for which these new methods are developed are expected to exhibit DTN-like characteristics with quasi-deterministic mobility[1], such that information travels between nodes in a store-and-forward manner, through paths that can be estimated with a certain degree of confidence in advance.

Much of the research in DTN routing is focused on opportunistic contacts, where estimation of future inter-node transfers is based on some probabilistic approach [1,40]. In satellite networks, however, mobility is quasi-deterministic, such that an idealistic contact plan can be prepared in advance, which is subject to uncertainty related to node availability on an ad-hoc basis [19]. In this environment, the most widely adopted approach is Contact Graph Routing [18], which is based on the schedule-aware bundle routing (SABR) framework [14]. In Nag et al. [32], SABR is used as the routing method for delivering data acquired by a constellation of satellites, working in combination with a locally-deployed Dynamic Programming (DP) scheduler that plans pickup operations for flood risk monitoring. Since on-board decision making is assumed, the delivery of tasks to their respective satellites is not considered. In other words, requests are not submitted for specific data pickup, tasks are generated on board based on the satellites' environmental knowledge. While minimising the latency of delivery for data generated within a DTN is typically the primary objective, a solution to maximizing data flow through DTNs is introduced in Li et al. [25], which could offer an interesting extension to this work.

Numerous enhancements from the baseline Contact Graph Routing (CGR) approach have been introduced since its original definition. For example, the issue of congestion (traffic load) is partially addressed in Madoery et al. [29] and Bezirgiannidis et al. [7] through tracking of local traffic demands and management of contact over-subscription, respectively. A mechanism to handle downstream contact uncertainty is provided in Raverta et al. [35], where a Markov Decision Process is used to provide a probabilistic approach to deriving routing decisions based on contact plan expectations. A centrally managed approach to path distribution is introduced in Caini et al. [13] and extended in Birrane et al. [9], which is of particular interest in this work. That extension, termed "Moderate Source Routing" (MSR), reduces the risk of route cycles forming, through the use of pre-computed delivery paths to route specific bundles. It is shown to be particularly effective where oracle-like knowledge exists at some central node. Other approaches to traffic awareness and congestion control, such as the MILP approach introduced in Fraire et al. [21] and the application of reinforcement learning in Silva et al. [38], offer interesting solutions with high performance, but either suffer as the network scales due to growth of the problem space and/or place additional computational demands on potentially resource-constrained remote nodes.

Despite the advances in nominal data routing performance using CGR, the route planning focus remains *reactive* with respect to the arrival of traffic, rather than being *proactive* in defining when, and where, data should enter the network. The work introduced here, offers this capability of engineering the traffic flow in such a way that fits to available network resources.

1.4. Article Contribution

The contribution of this work is the introduction of a novel method to allocate pickup and delivery tasks to satellites in space networks that exhibit delay-tolerant network behaviour and deterministic (or quasi-deterministic) mobility patterns. Triggered by the arrival of a request for goods, the output of the method is one or more tasks that are delivered to remote nodes, via one or more relays (transshipments), and provide instructions for the pick-up and delivery of data products. Requests are processed immediately upon arrival, with the objective of providing earliest delivery of the data to be acquired in order to satisfy the request. The approach ensures feasibility through consideration of the route taken by the task (i.e. for it to travel from scheduler to the pickup assignee), the route taken by the data post-pickup, and (optionally) network resource availability.

2. Problem Definition

The full problem life-cycle includes a data acquisition request being submitted by a customer to a scheduler node, triggering the creation of a task that details the data pickup requirements. Following pickup, the data is delivered to its destination. The time period between task creation and data pickup is referred to as the "Pickup Phase", and the time period between pickup completion and delivery is the"Delivery Phase". This process is illustrated in Fig. 1, which shows a highly simplified Earth observation satellite example.

The Gateway (or Ground Station) acts as both a scheduler that receives the request and the data destination, these need not be the same Ground Station but, for simplicity is, in Fig. 1. Connections between nodes are illustrated by vertical lines joining each node's respective horizontal line.

Consider the situation (Fig. 1) where two imaging satellites (*X, Y*) orbit the Earth and are capable of servicing customer requests through acquisition of images over certain locations. A request is submitted, by a customer, for an image to be taken of a particular location (the "Target"). Despite Satellite *X* being the first to fly over the Target, Satellite *Y* is designated as the *Assignee* as it offers the earliest opportunity for delivery. The optimal path, therefore, is highlighted by the bold line going from the Gateway → Satellite *Y* → Target → Satellite *Y* → Gateway. Now, consider hundreds of satellites, thousands of image requests, inter-satellite communication for data-relay, resource limitations, and a large network of ground-based gateways for tasking and data download, and the problem becomes difficult to comprehend, let alone solve. The use of a delay-tolerant network (DTN) architecture to represent the above-described scenario can formalize the problem. The reader is directed to Fraire et al. [18] for a thorough overview of much of the terminology used in the sections that follow.

3. Network Model

The network considered in this work is dynamic, with pair-wise contacts between nodes that are formed and lost at times that are known *a priori*, or can be predicted with a reasonable level of certainty. Exactly *how* this intermittent connectivity arises is arbitrary in terms of the methods being introduced, as it is the changing topology that is of interest, rather than the cause of it. That being said, in the satellite network application, contacts generally occur as a result of the relative position and attitude of satellites and ground nodes.

3.1. Contacts

A contact ($C_{ij}^{t_1,t_2}$), represents a unidirectional contact between two network nodes, i and j, with a start time t_1 (i.e. $C.start$)[2] and end time t_2 (i.e. $C.end$). Each contact is an opportunity for data transfer, during which data may, but are not required to, be forwarded from the contact's sending node i (i.e. $C.snd$) to its receiving node j (i.e. $C.rcv$). A contact has

[1] It is not necessary to have full knowledge of future mobility, but mobility estimation to a reasonable level of accuracy is required to develop contact schedules in advance

[2] "dot" notation is used to retain consistency with CGR definitions in Fraire et al. [18]

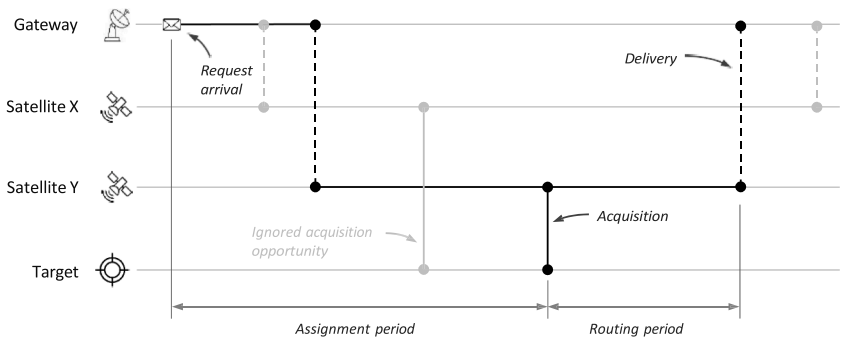


Fig. 1. Satellite pickup and delivery example. A single Gateway provides scheduling and task distribution, as well as acting as the destination to which the data must be delivered, while two satellites offer pickup opportunities from a Target node. Dashed lines represent gateway-satellite contacts, solid lines represent pickup opportunities, grey lines represent unused contacts. Time progresses from left to right.

a specific data transfer rate (*C.rate*) and, as a result has a maximum transfer volume capacity (*C.volume*) on the amount of data that could be transferred. Assuming a constant data rate, this volume is simply *C.rate* (*C.end*−*C.start*). Time taken for data to reach *C.rcv* after departing *C.snd*, i.e. the propagation delay, is termed the one-way light time (OWLT), *C.owlt*.

While it is not necessary to remove expired contacts from whatever Contact storing mechanism is being employed (e.g. a Contact Table), this is recommended as a way to reduce computational burden, unless required for future analytics or network behaviour predictions.

3.2. Contact Graph

A Contact Graph is a directed acyclic graph representation of a time-evolving network that captures connections between node-pairs occurring during some time horizon. Consider the Contact Graph CG = (*V, E*), where *V* is the set of vertices and *E* is the set of edges connecting adjacent vertices. Each vertex $C_i \in V$ represents a discrete Contact between two nodes, as described in Section 3.1. An edge $e_{ij} \in E$ connecting two Contacts C_i and C_j represents the sojourn period between the two contacts. That is, the period of time a node would store and carry data between adjacent contact opportunities in which it is the receiver of the first (during which the data is added to its local database) and sender in the second (during which it offloads the data and subsequently removes it from the local database). It follows, therefore, that a Node *m*, represented as the carrier of data on Edge e_{ij} is defined as $m = C_i.rcv = C_j.snd \; \forall \; i, j \in [0, \; |V-1|]$.

3.3. Routes

A Route R_A^B between two nodes, *A* and *B*, is represented by an ordered sequence of contacts (or *hops*) $\{C_0, C_1, ..., C_H\}$, where $C_0.snd = A$, $C_H.rcv = B$ and the number of hops is $H + 1$. For two adjacent hops to be considered feasible, it must be possible for data to reach the receiving node, after being transmitted from the sending node. That is, $C_i.start + C_i.owlt \leq C_{i+1}.end \; \forall \; i \in [0, \; H-1]$. Similarly, the sending node in a contact must equal the receiving node in the previous hop, that is $C_i.rcv = C_{i+1}.snd \; \forall \; i \in [1, \; H-1]$.

Each Route has additional attributes that can be derived from the properties of its contacts. A Route has a *Best Delivery Time* (*R.BDT*), which is the earliest opportunity for payloads to begin arriving at the final node. The *R.BDT* is more formally defined as the maximum across all contacts of their earliest arrival time (*C.arr time*) plus the time taken for the data to traverse the contact (*C.owlt*), that is $\max(C_i.arr\ time + C_i.owlt) \; \forall \; i \in [0, \; H-1]$, where $C_i.arr\ time = \max(C_i.start, \; C_{i-1}.arr\ time) + C_{i-1}.owlt$. This definition captures situations of contact overlap, where a later contact in the Route's hop sequence may begin prior to that of an earlier one, such that earliest arrival at the final node may come later than the start of the final contact. A Route also has a nominal volume, *R.volume*, that is the maximum amount of data it can carry from source to destination.

While nominal contact volume, as described in Section 3.1, is simply a function of the contact's transfer rate and start/end times, Route volume is dependent on the relative start and end times of its contacts with respect to each other, as well as their respective transfer rates and OWLTs. Fig. 2 provides an illustration of how contact overlap can impact Route volume and best delivery time. The reader is directed to Fraire et al. [18] for a full definition of how volume is modelled in Contact Graph routes, which goes into more detail than is considered necessary for the purposes of this work. Note that the contact "blocks" in Fig. 2 indicate times during which bundles (i.e. the Bundle Protocol specific term used for data packets at the BP layer) can be "sent", rather than "received", hence why the fourth bundle is shown as arriving *after* the final contact ends.

3.4. Requests

Submission of a request, *Q*, for pickup and delivery is the mechanism by which the Contact Graph Scheduling process is initiated. A request includes identification of the Target endpoint identifier (EID) *Q.target* from which the data must be collected and the Destination EID *Q.dest* to which the data must be delivered. Note that the Target EID and/or Destination EID may reasonably map to more than one Node. For example, delivery of the data to any one (*G*) of a set of gateways (G) may

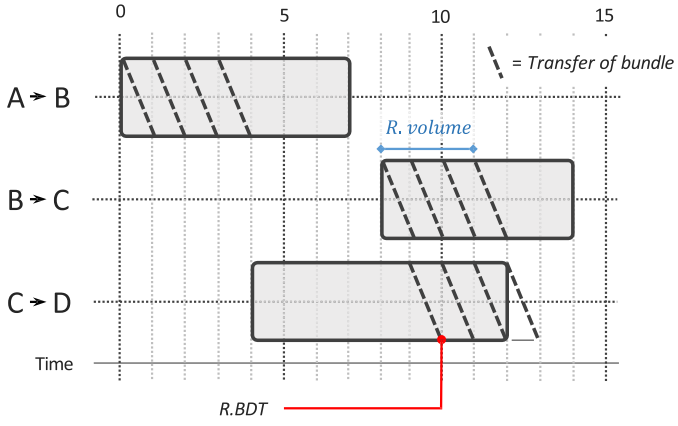


Fig. 2. Illustration of a route from node A to D. Diagonal dashed lines represent the movement of data (i.e. bundles, in this context) from one node to another, during a contact. In this example, the one-way light time (OWLT), or propagation delay, is 1 time unit for each contact, and the time between the transmission of each bundle is also 1 time unit. Given overlap between contacts $C_{B,C}^{8,14}$ and $C_{C,D}^{4,12}$ the route can accommodate 4 bundles, with a best delivery time (*R.BDT*) of 10.

be acceptable, such that these gateways would all share an EID. That is, in the destination scenario, $G_k.EID = Q.dest \; \forall \; G_k \in G$. A pickup deadline (*Q.pu*) may be defined, as can a delivery deadline (*Q.del*) by which the data must reach its destination and/or a maximum time to live (*Q.ttl*) in case data age is important.

Inclusion of some limit on delivery time is recommended, as a way to limit the time horizon over which the search must be carried out, but is not strictly necessary. Finally, a list of "excluded" nodes that are forbidden from acquiring, relaying or delivering the data (*Q.excl*) may be included without negative impact on the method, but are not considered in this work for reasons of clarity. Data volume (*Q.vol*) may also be defined as part of the request, if it is not implied by some other method. This must account for any overhead required to encapsulate the payload data within one or more bundles (Section 3.6), such that resource consumption modelling is accurate. Regarding data priority, requests are handled on a first-come first-served basis, such that arrival time defines priority in relation to scheduling. Once resource has been assigned during the scheduling process, within the Scheduler's internal network model, it is assumed that this resource cannot be re-allocated in response to a later request. Note that this resource allocation may not actually be realised during the routing of data to its destination, since the allocation of data packets along specific routes is the responsibility of the local routing mechanism, e.g. CGR.

3.5. Tasks

For every Request that is deemed feasible, i.e. requests for which pickup and delivery is expected to be completed prior to their respective deadlines, a Task object (*T*) is created. As a minimum, a Task must include details of the Target EID from which the payload data is to be picked-up, the destination EID to which the data must be delivered, the deadlines for pickup and delivery and the node assigned to execute the pickup (*T.assignee*). Additional (optional) attributes include the specific time of pickup (*T.pu*) and the route along which the data would ideally travel once acquired. If no feasible pickup or delivery opportunity exists, however, a task should not be generated since it would be a waste of resources. This would be the case if, for example, there are no acquisition opportunities prior to *Q.pu* or valid delivery opportunities prior to *Q.del*.

Each Task object has a status attribute (*T.status*), which takes one of the following values; "pending", where the data is yet to be picked up, "acquired", where the associated data has been collected, but not yet delivered, "delivered", where the associated data has successfully been delivered to its destination, "failed", where the task is no longer able to be completed before either the acquisition or delivery deadlines, whichever is relevant to its current state, and "re-scheduled", where there is no possibility for the current task to be completed, but a new, equivalent, task has been generated with potential to be completed. A Task would transition to the "re-scheduled" state, if for example a dedicated pickup operation was missed by the assignee, but sufficient time remains prior to *Q.pu* for pickup by the same, or different, node. In some situations, it may be more appropriate to edit an existing Task, rather than creating a new one, however care would need to be taken to ensure any Task conflicts (due to the existence of multiple, different versions of the same Task in circulation at one time) are handled. This issue is considered an implementation matter, and is not critical to the methods being introduced here. Transition between these various states can be described in terms of a finite state machine (Fig. 3). Note that "delivered", "failed" and "re-scheduled" are final states, from which no further activity can take place, and "pending" acts as an initial state for a Task.

In this work, Task objects are assumed to be negligible in size, such that they can be flooded through the network without significant impact on contact capacity. As such, it is assumed that tasks are shared between Nodes as part of a *handshake* operation, in which each node updates their set of tasks based on the most up to date information from both parties. While flooding maximises the probability of Tasks reaching their *Assignee* in good time, in situations where the resources consumed by Tasks is non-negligible, Tasks could be delivered to their respective *Assignee* using CGR (or some other routing method) without any loss of generality. The Task *Assignee* EID would, in this case, act as the Task destination in the routing process.

3.6. Bundles

The result of a successful pickup operation is the introduction of data to the network, for example a message, or image file. Once acquired, this data is encapsulated into one or more bundles (as defined in the Bundle Protocol Burleigh et al. [12]) so that bundle-specific properties can be exploited during the delivery phase. To avoid confusion, whenever discussing transfer of "data" through the network, this refers to bundle/s that store the files (e.g. an image) associate with a specific request. Each bundle (*b*) includes a destination endpoint (*b.dst*), which maps to one or more nodes to which the bundle can be "delivered", a level of priority (*b.p*) that can be used to order bundles for routing precedence, and a size (*b.size*), which indicates the amount of network resource consumed by the bundle during storage and node-transfer (this equates to the requested data volume, *Q.vol*). Other attributes may exist on the bundle, such as a

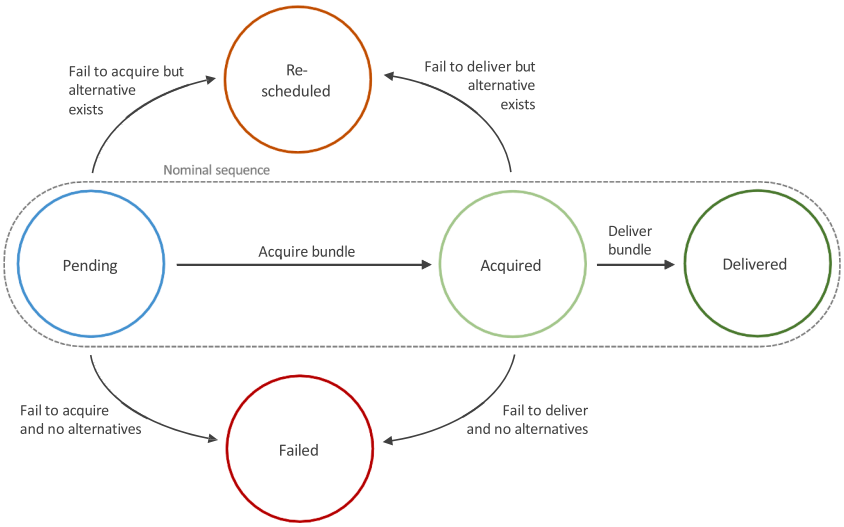


Fig. 3. Finite state machine diagram for a Task object

list of nodes to which the bundle should not be forwarded (*b.excl*), an expiry date after which the bundle is considered to have zero value and should therefore be dropped from the network (*b.deadline*) and a pre-defined route over which the bundle should be delivered, if possible to do so (*b.route*).

4. Contact Graph Scheduling

Contact Graph Scheduling (CGS) is a deterministic, shortest-path approach to solving an adaptation of the dynamic pickup and delivery problem with time windows and transshipments. CGS identifies the network nodes best suited to execute tasks based on some request-specific objective, which in this article is minimising the time taken from request arrival to data delivery. Constraints must be adhered to, such as pickup and delivery prior to their respective deadlines. An optional feature of CGS is the consideration of network resource consumption during the task generation process, as a way to decrease the risk of network congestion through contact over-subscription. Here, the consumption of contact transfer capacity is considered, as a reasonable first look at the impact of resource-awareness.

Note that Contact Graph Scheduling *is not* a method that seeks global optimality. Relaxing the search for globally optimal solutions, by handling requests as they arrive (rather than as a collection), offers request-specific service-level guarantees such that infeasible requests can be declined immediately, or negotiation can be made as to the request performance. As such, CGS is not presented as a competitor to the traditional linear programming approaches to satellite scheduling, but instead offers an alternative method that lends itself well to applications needing low computational demand, rapid request feedback and the potential for decentralized deployment on-board resource-constrained platforms.

The CGS procedure is illustrated in Fig. 4 and defined in detail in the following text and algorithm (Algorithm 1). The arrival of a request Q initiates the process, which aims to find the path that travels via the request target ($Q.target$) and reach the request destination ($Q.dest$) at the earliest opportunity. These can be considered two separate paths, the "acquisition path" from source (S) to target (T), completed during the the *pickup phase*, and the "delivery path" between the target and destination (D), completed during the *delivery phase*. It is a necessary condition for at least one pair of acquisition:delivery paths to be identified, prior to the generation of a task. In other words, for a request to be accepted, it must be possible to both pickup the associated data prior to the acquisition deadline and deliver that data prior to its expiry. Contact opportunities with the request target location ($C_{ij} \in$ C: $j = Q.target$) are computed either *a priori*, if possible, or in response to request submission. For example, the contact opportunities associated with a request for satellite image acquisition over a specific location on Earth, will likely need computing in response to the request submission, using an orbital propagation model.

Algorithm 1
Contact Graph Scheduling

Data: source S, target T, destination D, contact plan CP, request Q
Result: pickup path R^*_{pu} delivery path R_{del}
1 Clear(CP) // reset all contacts
2 $EDT = \infty$ // preset the earliest delivery time
3 $R^*_{pu} = \text{None}, R^*_{\text{del}} = \text{None}$ // preset the selected paths
4 **while** True **do** // Iterate over the CGS main loop
/* find the shortest pickup path */
5 $R_{pu} = \text{DIJKSTRA}(C^{t,Q.pu}_{S,S}, T, CP, Q.pu, 0)$
6 if R_{pu} is None then // if no acquisition path found, exit
7 break
8 if $R_{pu}.bdt \geq EDT$ then // if pickup later than EDT, exit
9 break
10 A $= R_{pu} \cdot hops[-1].frm$ · // identify the assignee
/* Suppress all contacts between the assignee and the target */
11 **for** $C \in CP$ **do**
12 **if** $C.frm =$ A & $C.to = T$ **then**
13 $C.suppr = TRUE$
/* find the shortest delivery path from this pickup event */
14 $R_{del} = \text{DIJKSTRA}(C^{R^{pu}.bdt, R^{pu}.bdt}_{T,A}, D, CP, Q.del, Q_{vol})$
15 if R_{del} is None then // if no valid delivery paths, skip
16 skip
17 **if** $R_{del}.bdt < EDT$ **then**
18 $EDT = R_{\text{del}}.bdt$ // update the earliest delivery time
19 $R^*_{pu} = R_{pu}, R^*_{\text{del}} = R_{\text{del}}$ // assign "selected" paths

4.1. Algorithm Overview

A pseudo-code implementation of the CGS process is provided in Algorithm 1, making use of the notation already introduced in previous sections and the Contact Graph implementation of Dijkstra's algorithm introduced in [18]. The CGS process begins by resetting all contacts in the Contact Plan (line 1), which involves resetting Contact attributes related to shortest path discovery that may have been set during a

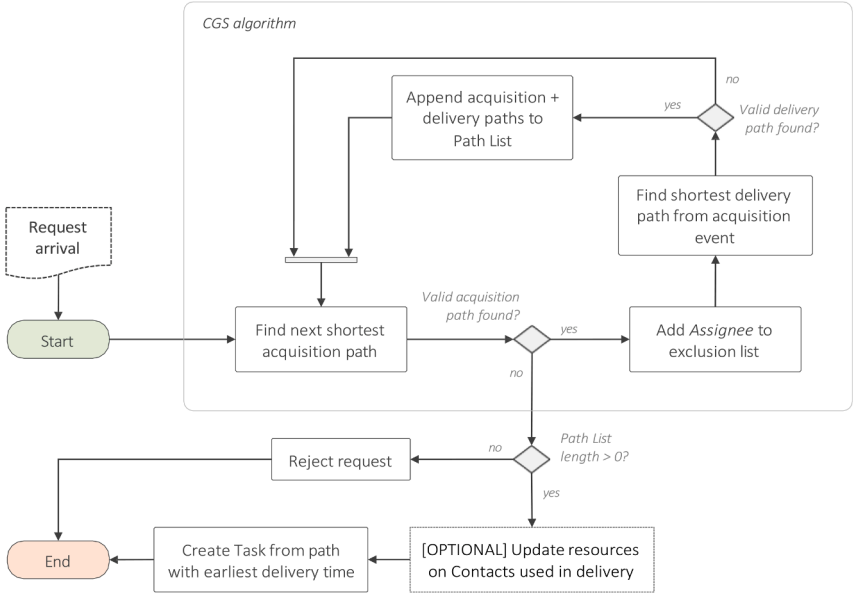


Fig. 4. Flow-chart illustration of the Contact Graph Scheduling process

previous execution, and initialising the *earliest delivery time* (*EDT*) (line 2), selected pickup route (R^*_{pu}) and delivery route (R^*_{del}) (line 3). Then, the discovery process begins (line 4) with identification of the earliest arriving pickup path (line 5). That is, the shortest path that concludes at a contact with the target node (*T*), which can be considered the moment at which the requested (payload) data would be acquired. It is assumed that Dijkstra's algorithm is used, but any shortest path algorithm would suffice. If no valid pickup path exists, the process can be terminated (lines 6-7). If a pickup path *is* identified, but its arrival at the Target is later (or equal to) the current *EDT* (line 8), the process is also terminated (line 9). If the arrival at the target is earlier than our current *EDT,* there's a chance we will find an improved complete solution, so the Assignee A can be identified as the sending node of final hop in R_{pu} (line 10). This is the node that makes contact with the Target and, therefore, acquires the payload data. Lines 11-13 provide the procedure to remove any later contacts between the Assignee and the target, from the Contact Plan, since they will add no improvement over the pickup path already discovered and potentially result in unnecessary computations during future iterations. At this point, it is necessary to identify the best delivery path (R_{del}), considering a starting point for this search as the *BDT* of the pickup path, since that is the earliest time at which the payload data is received by the Assignee from the Target. R_{del} can also be discovered through execution of a shortest path algorithm (line 14), considering the payload data size (Q_{vol}) as the minimum route volume required to successfully traverse this path. If no feasible delivery path is found from the current pickup opportunity (line 15), the procedure moves on to the next CGS iteration, which will investigate the *next* best pickup path. In the event that a valid delivery path *is* found, the best arrival time at the Destination should be checked against our current *EDT* (line 17) and, if an improvement, update the *EDT* (line 18) and assign both this delivery path (R_{del}), and the associated pickup path (R_{pu}) to R^*_{del} and R^*_{pu}, respectively.

The Dijkstra algorithm implemented at lines 5 and 14, is similar to the Contact Graph Routing implementation as described in Algorithm 1 in Fraire et al. [18], with the first three arguments being the root contact from which the search begins, the destination node that must be the receiving node on the final contact, and the full Contact Plan. Two additional arguments are included for CGS completeness, which are the deadline by which a feasible route must be complete (*Q.pu* for pickup and *Q.del* for delivery) and the volume of data to traverse the route (*Q.vol*). In other words, this sub-routine returns the ordered sequence of contacts capable of accommodating the task/payload traversal, offering earliest arrival at the target/destination node, respectively.

The CGS algorithm provides recommendations for both the pickup path (R^*_{pu}) and delivery path (R^*_{del}), returning null values in the case of an infeasible request. If valid routes are returned, a task should be generated for distribution through the network. Failure to identify a valid pickup or delivery path could be due to either unavailable resources or a lack of feasible route to the Assignee or Destination prior to the respective deadline.

The creation of a task, based on the outputs from the CGS algorithm, can, if deemed necessary, be followed by modification of network resources that would be consumed by the bundles traversing their delivery paths. Doing so reduces the risks of contact over-subscription from later requests (requiring re-routing) and the need to re-schedule tasks that fail to be completed. If implemented, it should include the reduction of contact bandwidth for each hop, $C_i.vol = C_i.vol - R.size \forall i \in R^*_{del}.hops$, but could also include the consumption of on-board storage and/or energy available for data transfer. For the purposes of this work, storage capacity and energy are considered to be infinite. Maintaining this log of Contact capacity is of particular benefit when there is a central node executing all scheduling operations, since it has full oversight of idealistic traffic flows. In the distributed scheduling case, however, resource consideration would be challenging to communicate through the network due to intermittent connectivity generally preventing such real-time information sharing, such that each node's opinion on the network resource state would be unique to them.

Whether or not the pickup and delivery paths are adhered to in practice, is an implementation matter. Indeed, in the case where no uncertainties exist and all contacts occur as expected, following the CGS-defined paths would have benefits. Specifically, adhering to the suggested paths will make good use of the traffic-flow knowledge held only at the Scheduler node. On the other hand, local routing may send bundles along paths that become over-subscribed. In order to exploit traffic-flow knowledge from the Scheduler node, Moderate Source Routing (MSR) [13] can be used for bundle delivery, such that bundles are assigned a route that should be followed if possible to do so. In reality of course, where uncertainty exists and contacts may not occur always as expected, even use of MSR for bundle delivery cannot prevent occasional failure to traverse preferred routes. If MSR is not used, or the preferred route defined using MSR is no longer feasible, it is recommended that CGR is used instead.

4.2. Algorithm Time Complexity

Contact Graph Scheduling consists of potentially multiple iterations of pairs of shortest paths being computed, one for the pickup phase and one for the delivery phase. Each of these computations can be achieved through the application of Dikjstra's algorithm, which has a worst-case time complexity of $O(|C|log|C|)$ in its Contact Graph implementation [18]. The number of iterations is limited to, at most, the number of unique nodes ($|M|$) in the network available for carrying out acquisition operations, since a later acquisition from the same node will offer no improvement on an earlier opportunity. Therefore, a worst case time complexity, ignoring constant scale factors, is $O(|M||C|log|C|)$.

In reality, performance will likely be significantly better than this. Algorithm exit conditions dictated by acquisition and delivery deadlines can reduce the time horizon scope dramatically, and the use of pre-computed contact-list hash tables can reduce the need to explore *all* contacts within the Dijkstra implementation.

4.3. Assumptions and Limitations

A number of assumptions have been made in this implementation of CGS on a delay-tolerant satellite network. First, the assumption that buffer storage capacity is infinite is common in DTN route discovery, since consideration of this attribute would have knock-on implications on routes elsewhere in the network. While data transfer rates are increasing rapidly due to advances in technologies such as optical (laser) communication, it remains the case that contact transfer capacity is typically smaller than data storage capacity. This assumption would of course need to be addressed in some terrestrial applications, where data storage is more likely to be the limiting factor, rather than volume of transfer during transshipment events, as is the case for traditional logistics and vehicle delivery scenarios. Second, it is assumed that once network resources are allocated for the pickup and delivery of data, that resource cannot be re-assigned during later request processing. While this may not be preferable when considering different levels of data priority, it offers a clear baseline on which improvements can be made. The assumption that task information cannot be continuously updated on all nodes further supports this assumption, since reallocation of resources should be communicated to the affected nodes, which may be difficult depending on the specific Contact Plan. Again, in terrestrial applications, it may be more reasonable to assume continuous, real-time communication with vehicles, such that changes to resource allocations is easily distributed.

5. Analysis

A discrete event simulation (DES) has been constructed to model the operational aspects of a customer-scheduler-satellite-gateway network

and to evaluate Contact Graph Scheduling performance. This model includes submission of requests, scheduling of tasks, distribution of task information through the network and the pickup, relay and delivery of data to their respective destinations.

5.1. Scheduling Schemes

Five approaches to scheduling incoming requests are compared, each offering a higher level of sophistication than the last, at the expense of additional computational demand. The details of each approach is presented in Table 1 with further discussion around their capabilities in the following paragraph. Three of the schemes are built on top of CGS, with the consideration of resources and definition of delivery paths being the differentiating factors separating them.

The **"Naive"** approach is simple in that it creates a Task object for every Request that is submitted, regardless of pickup/delivery feasibility or resource availability. No assignee is defined on the task, such that any node with this Task in a "pending" state will pickup the requested data if they come into contact with the Task's target node. This scheme will result in relatively high network congestion due to multiple copies of Task-specific bundles in circulation. The **"First"**approach identifies the *first* node capable of completing the pickup operation and assigns the task to that node. This is achieved through a single execution of Dijkstra's algorithm on the Contact Graph, as per line 5 of Algorithm 1. No consideration of the delivery or resource availability is included, such that bundle delivery feasibility is not guaranteed. The positive impact of this pick-up assignment is that no more than one instance of the requested data will be generated for each task. The **"CGS (PU)"** ("PU"standing for "Pick Up") scheme makes use of the CGS procedure to identify the acquisition opportunity that minimizes the time from request submission to bundle delivery, but does not prescribe a specific delivery path to be used, nor does it account for resource consumption during the delivery phase. This maintains simplicity in the sense that the Contact Plan is not required to be continuously updated based on resource consumption, but risks assigning tasks to nodes that later result in delivery failure due to congestion. The **"CGS (CGR)"** approach implements all aspects of the Contact Graph Scheduling method, including resource consumption, but *does not* force the use of the identified delivery path. This implements CGR for bundle routing. Finally, **"CGS (MSR)"** implements the same scheduling approach as *CGS (CGR)*, but defines a delivery route for each bundle that must be adhered to if physically possible. Remote nodes then use Moderate Source Routing (MSR) during the delivery phase to maximize the probability of routing bundles around network bottlenecks. If, for whatever reason, a bundle is unable to follow its pre-defined route, CGR will be used for routing.

5.2. Scenario Definition

A realistic space network scenario is used during the analysis, with a topology equivalent to that introduced in [20]. This includes a 16-satellite, 4-plane Walker Delta constellation [41], a set of six ground-based gateways for task distribution/data delivery, and 25 target nodes distributed across the globe. The satellites are in circular, 500km altitude orbits, at an inclination of 50°, with an inter-satellite link (ISL) range of 1000km. Inter-satellite separation distance is the only factor defining contact opportunities, i.e. omnidirectional antennae are assumed for both transmission and reception. Pickup from Targets is considered possible when the satellite is greater than 30° above the Target node's horizon, and data delivery is possible at elevations of greater than 10° above a Gateway's horizon. Data products are considered "delivered" if transferred to any one of the six gateway nodes.

Table 1
Scheduling schemes used for comparison in the analysis. Note that "Valid Pickup?" and "Valid Delivery?" columns indicate whether or not a valid pickup or delivery opportunity must exist, for the request to be accepted. For example, in the *Naive* case, neither valid pickup nor delivery need be identified, in order to accept a request

Scheme Name	Valid Pickup?	Valid Delivery?	Resource aware?	MSR used?
Naive	No	No	No	No
First	Yes	No	No	No
CGS (PU)	Yes	Yes	No	No
CGS (CGR)	Yes	Yes	Yes	No
CGS (MSR)	Yes	Yes	Yes	Yes

The model begins with propagation of the satellites, considering perturbations from J_2 Earth oblateness, using the Gaussian form of Lagrange's planetary equations of motion [42], with fixed time steps of 10 seconds. A contact schedule is then derived by evaluating relative positions between satellite and terrestrial nodes, storing the start and end times of possible pair-wise connections. Owing to the nature of this constellation configuration (i.e. a low Earth orbit Walker Delta constellation), duration of a typical space-ground contact is on the order of 3-8 minutes and for a space-space contact is approximately 4 minutes. A discrete event simulation (DES) is then executed, during which requests arrive to a Scheduler node, are converted into Tasks (if deemed feasible) and distributed through the network. Data pick-up, inter-satellite relay, and delivery is carried out according to Task instructions and/or local routing decisions made on board. For example, if it is determined by a satellite that one or more particular bundles should be transferred to a neighboring node during a particular contact, transfer of those bundles will be realised within the simulation at the earliest convenience during that specific contact event.

Request arrival follows an exponential distribution, with a expected inter-request arrival time[3] ($T_{request}$) of

$$Trequest = \frac{t_{sim} b.size}{\lambda C_d}, \tag{1}$$

where t_{sim} is total simulation time, $b.size$ is the size of the bundle/s that are generated for the data pick-up in response to each request, C_d is the total data download capacity from all satellites over the simulation period, and λ is the "request submission load" (RSL). Note that this RSL value represents the intended ratio between data upload and potential download over some time horizon. However, since it is not guaranteed that all requests are deemed serviceable, the realized traffic load (congestion) may be lower than this target value. For example, if requests included an unreasonably early pick-up and/or delivery deadline, such that acquisition and/or delivery was deemed infeasible in many cases, those requests would be rejected and that traffic wouldn't flow through the network. This presents an interesting observation that is not typically seen in data routing analyses, such that requests can be rejected upon arrival (if deemed infeasible), or fail having previously been processed into a Task. This latter state could be seen as a proxy for customer dissatisfaction regarding some service level agreement, since their accepted request was never converted to a successful data product delivery.

The discrete-event simulation runs from an epoch of 12:00pm (UTC) on 1st January, 2023. A warm-up period of 3 hours is considered, during which traffic flow is allowed to build up. This warm-up period mitigates against the impact of including data traffic flow results while the network is yet to reach a steady-state, i.e. while congestion is artificially low. Data flow, corresponding to all requests submitted during the 10 hour period immediately following this warm-up period, is monitored until either delivered or dropped from the network. These request-data pairs are used in the calculation of network performance shown in the results sections below. Request submission, and the associated bundle traffic flow is allowed to continue beyond this 10 hour monitoring

[3] Expected inter-request arrival time is the mean average time between request arrivals, as measured over the long-term

period, despite their delivery metrics not contributing to the results. This is done to ensure routing behaviour is not adversely affected from an unrealistic reduction in network congestion. Monitoring, on any remaining tasks and bundles that relate to requests received during the active period, is continued during the cool-down period. This cool-down period ends when all traffic related to requests that arrived during the active period, has been either delivered or dropped. This process is illustrated in Fig. 5.

The code used to generate the results presented below can be obtained at Lowe [27], and the code repository for this work can be found at Lowe [26]. A summary of all necessary network parameters are provided in Table 2.

Table 2
Space network simulation attributes

Attribute	Value	Units	Description
Simulation			
Warm-up	3	hours	Period for traffic flow to build
Duration	10	hours	Active monitoring period
Start date	01/01/23	-	Start date of the simulation
Start time	12:00	-	Start time of the simulation
Traffic			
Bsize	100	MB	Size of a single bundle
TTTL	3	hours	Maximum Task lifetime
BTTL	2	hours	Maximum data (bundle) lifetime
Communication			
RISL	100	MB/s	Inter-satellite data transfer rate
RS2G	100	MB/s	Satellite-to-Gateway data transfer rate
DISL	1,000	km	Max inter-satellite link range

5.3. Performance Metrics

To illustrate the performance of the different scheduling schemes modelled, a number of performance metrics are presented, including:

a) **Total Latency**; the mean average time from request submission to data delivery, for all delivered data.
b) **Pickup Latency**; the mean average time from request submission to data pick-up, for all delivered data.
c) **Delivery Latency**; the mean average time from data pickup to data delivery, for all delivered data.
d) **Request Success Rate**; the ratio of the number of requests that had at least one data instance (could be multiple, in the Naive case) successfully delivered to the total number of requests submitted.
e) **Accepted request success Rate**; the ratio of the number of tasks (i.e. *accepted* requests) that had at least one data instance successfully delivered to the total number of tasks generated.
f) **Hop Count**; the average number of hops (node transfers) taken by bundles that were successfully delivered.
g) **Requests Accepted**; the total number of requests that were deemed feasible and, thus, converted into tasks.
h) **Requests Failed**; the total number of requests that were deemed feasible, but never resulted in a delivered data instance. I.e. they "failed" after being accepted
i) **Requests Delivered**; the total number of requests that resulted in at least one data instance being delivered (i.e. the number of "fulfilled" requests)

Owing to the inter-connected nature of how performance can be measured in scheduling and routing applications, it is important to consider information together and not draw significant conclusions from a small sub-set of results. For example, the same low latency performance can be achieved through either efficient scheduling/routing or by dropping a high proportion of bundles. The latter is clearly less attractive when considering customer satisfaction.

5.4. Results and Discussion

For each of the scheduling schemes presented in Section 5.1, a variety of request submission loads (RSLs) are evaluated. Recall that a lower request submission load corresponds to a lower data traffic load in the network (assuming all requests are accepted), and thus lower traffic congestion. An RSL of greater than 1 is considered reasonable (and therefore evaluated), if for example the service was popular to the point that requests arriving from 3rd parties exceeded the capability of the network. In this case, it is the responsibility of CGS to identify this potential congestion issue, and reject requests that will knowingly result in over-booking and failure of delivery. For each specific RSL value, an identical request arrival sequence is used across each of the routing schemes, to ensure equivalent comparison. In the case of any of the three CGS schemes, infeasible requests should be rejected at source, rather than converted into Tasks that are inevitably dropped later. In the Naive approach, since it is possible for multiple instances of the data to be generated for a single request, traffic flow will likely *exceed* RSL, resulting in high levels of network congestion.

5.4.1. Scheduling under Nominal Conditions

Results for each of the performance metrics, measured against request submission load, for each of the scheduling schemes, are presented in Fig. 6. Results are generated from a single simulation, based on warm-up, monitoring and cool-down timings as shown in Fig. 5.

In terms of latency (Fig. 6a, b, c), it can be seen that the more sophisticated methods (green and yellow) result in longer latency overall, with the *CGS (MSR)* having the greatest average time from request submission to payload delivery (Fig. 6a). While, in the first instance, this may seem like an unsatisfactory result, when combined with the fact that a greater number of requests are delivered (Fig. 6i), the benefits become clearer. Indeed, the requests that are accepted (i.e. ones for which a task is issued) and not delivered, fail because at least one of their associated bundles either have or would have (had it not been dropped prior) remained in transit beyond its delivery deadline. These failed requests (Fig. 6h) do not contribute to overall latency and their omission is significant due to the typically long latency they would have exhibited had they remained in transit. That is, in the schemes where the, otherwise failed, requests *are* successfully delivered, they are a major contributor to that scheme's total latency results. This effect is amplified

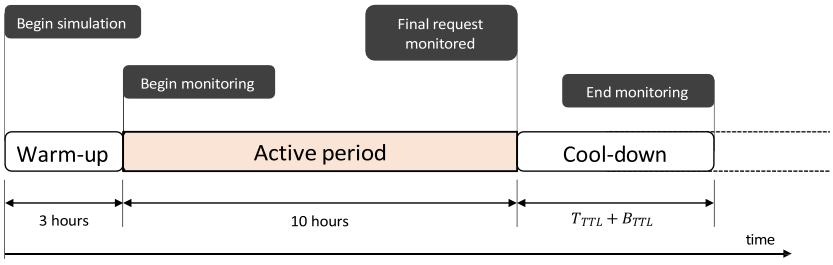


Fig. 5. Outline of the simulation phases, including a warm-up phase to introduce traffic into the network, an active phase, during which monitored requests arrive and a cool-down period during which any remaining traffic activity associated to requests that arrived during the active period, are monitored.

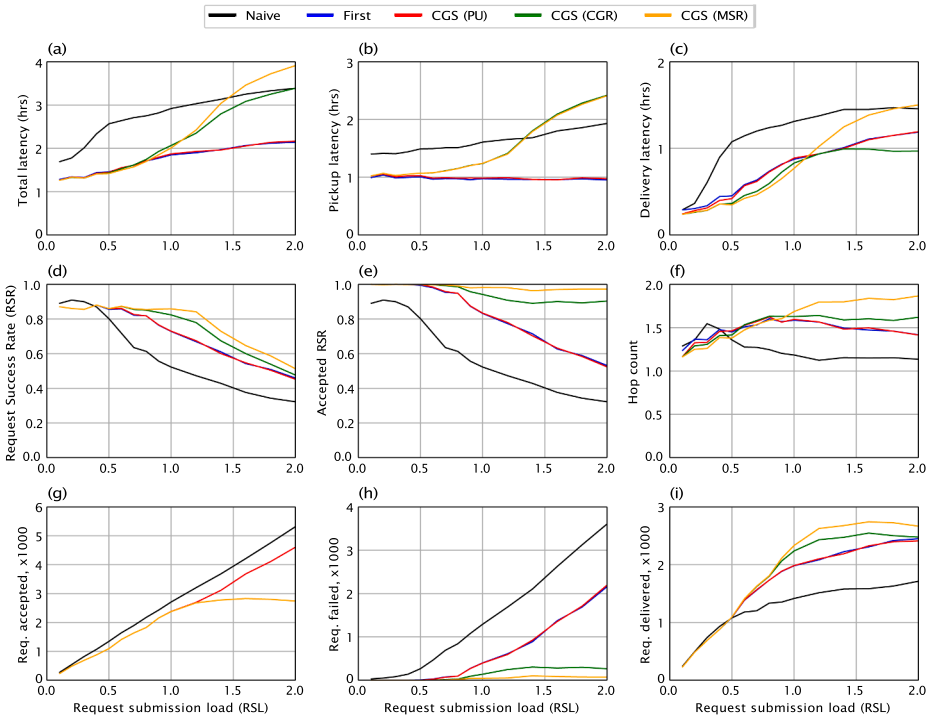


Fig. 6. Results of a 10 hour simulation for a 16-satellite, 25-target, 6-ground gateway scenario. Each performance metric (y-axes) is plotted against incoming request submission load. Note, in figure **g**, the *First* scheme results (blue) lie directly under the *CGS (PU)* results (red).

in the case of the Naive approach, since the number of aging bundles increases due to network congestion blocking their delivery. One interesting result is the lower delivery latency (Fig. 6c) in the *CGS (CGR)* approach, at higher RSLs in comparison to the *CGS (MSR)* approach. This can be attributed to the fact that beyond an RSL of 1, where it would be impossible to deliver all requests, an increase in failed requests beyond that of its MSR counter-part begins to be seen. MSR, on the other hand, is demonstrating its value, by exploiting the knowledge generated at the Scheduling node and deliberately routing bundles along higher latency paths rather than taking shortcuts that sometimes end in failure due to over-crowding. This finding is supported by the Hop Count plot (Fig. 6f), in which MSR generally routes bundles on longer journeys in terms of hop-count, as a way to bypass these network bottlenecks.

A benefit of resource awareness and consideration is evident in the plots for Requests Accepted (Fig. 6g) and Requests Failed (Fig. 6h), which can be considered representative of the customer-scheduler interaction. The *Naive* approach, by definition, accepts *all* submitted requests, and therefore results in a significant proportion of customer dissatisfaction, with a high failure rate (Fig. 6h). The approaches that check for pick-up feasibility, but don't consider network resource consumption (blue and red), accept fewer requests, but once congestion increases beyond what is technically feasible from a data transfer perspective, they fail to spot this and continue to accept requests that would later be rejected by the network. Both resource-aware approaches (green and yellow) begin rejecting an increasing number of requests at this point (RSL $\approx$ 1.2), avoiding later disappointment (and the unnecessary traffic congestion these bundles would bring). In particular, the use of MSR in the delivery phase works well to minimize request failure, resulting in the highest volume of delivered payloads (Fig. 6i). The Request Success Rate plot (Fig. 6d) is perhaps a more intuitive measure of overall "fulfillment" as it shows the ratio between delivered and submitted requests, or in other words a measure of submitted request success rate. The Accepted Request Success Rate (Fig. 6e) takes this a step further, showing the ratio between the number of delivered requests and accepted requests (or tasks created), or in other words a measure of the accepted request success rate.

The similarity between the *First* and *CGS (PU)* schemes further illustrates the importance of resource consideration, at least in this particular scenario. Despite the *CGS (PU)* scheme identifying a task assignee according to minimum total latency (including a review of the data delivery path), the similarity in performance to an equivalent scheme that does not make this consideration, indicates that the first acquisition opportunity typically offers the lowest latency route in an infinite resource scenario. However, as can be seen from the divergence in pickup latency curves (Fig. 6b) on the *CGS (CGR)* and *CGS (MSR)* schemes, from RSL of ~0.5, the first acquisition opportunity becomes *less* likely to offer minimum total latency, due to pre-existing traffic flows.

5.4.2. Scheduling Under Contact Uncertainty

Uncertainty, in this work, is modeled as a probability of failed contact opportunity, specifically in the context of task and data transfer between satellites and/or gateway nodes. In the satellite context, this can be considered analogous to a lack of communication, which may be due to failure from an under performing sub-system, a lack of on board energy, conflicting attitude requirements or high computational load being demanded for other tasks.

Results shown in Fig. 7 illustrate results from the same operational scenario as in the previous section, but with a contact reliability of 70%. A number of differences are evident, resulting from the decrease in contact reliability. The dominant effect is that of an increase in request failure (Fig. 7h) across all scheduling schemes. In particular, while the *CGS (CGR)* and *CGS (MSR)* schemes demonstrated request failure rates of 9.6% and 2.7% under nominal conditions, respectively, this rises to 35.2% and 40.6% when contacts were unreliable. The in-creased level of failure rate in the *CGR (MSR)* case can be attributed to the generally longer routes that are defined in advance, which avoid network bottlenecks, but when failed present a greater risk of bundle expiration. There is also a noticeable reduction in Request Success Rate and Accepted Request Success Rate (Fig. 7d and e) across all schemes, even at low levels of congestion, suggesting that it would be prudent to apply a certain service-level guarantee relating to request fulfillment under

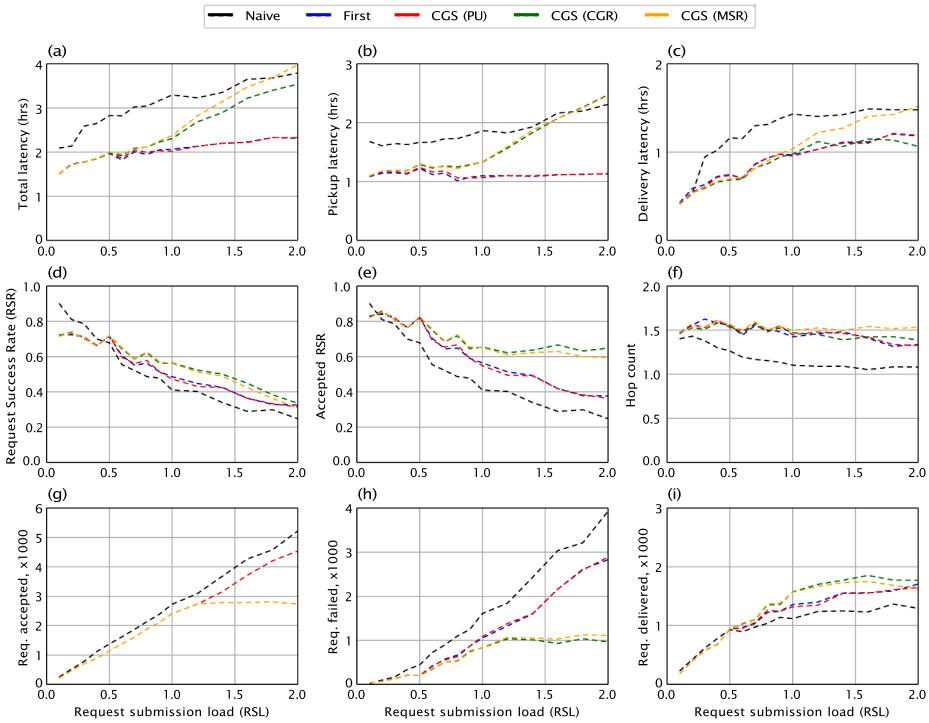


Fig. 7. Results of a 10 hour simulation for a 16-satellite, 25-target, 6-ground gateway scenario. Each performance metric (y-axes) is plotted against incoming request submission load. Contact uncertainty (i.e. the probability that a contact will be available for bundle transfer use) is 0.7. Note, in figure **g**, the *First* scheme results (blue) lie directly under the *CGS (PU)* results (red).

uncertain conditions, even for those requests that have been nominally accepted.

Results shown in Fig. 8 illustrate the behaviour of each performance metric, for the *CGR (MSR)* scheduling scheme, at different link reliability ranging from 0.7 (i.e. a 70% probability of a contact being realized) to 1.0 (i.e. idealistic conditions). All other simulation attributes remain as per the results in Section 5.4.1.

At low levels of request submission load, latency is generally higher (Fig. 8a), both for acquisition (Fig. 8b) due to some tasks failing to reach their assignee before time of acquisition, and delivery (Fig. 8c) due to

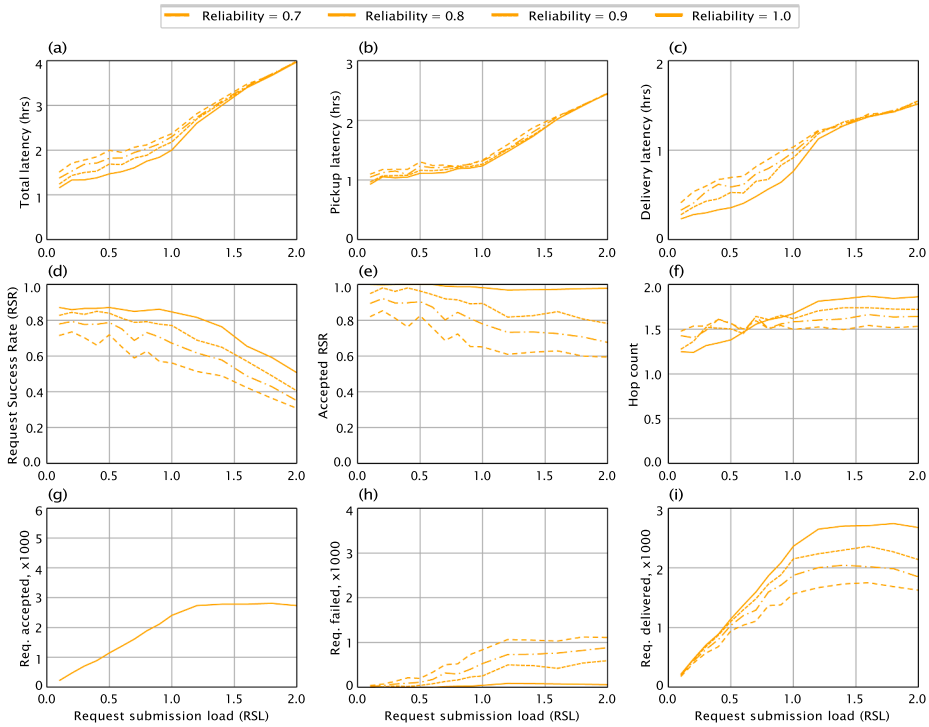


Fig. 8. Results for the full *CGS (MSR)* implementation on a 10 hour simulation for a 16-satellite, 25-target, 6-ground gateway scenario, with varying levels of contact uncertainty. Reliability represents the probability a data transfer contact is executed, such that lower values represent a higher number of failed contact attempts.

bundles failing to follow their intended path to the destination. At high congestion, the number of failed requests increases with decreasing reliability, which is to be expected, with a reduction in delivered requests (Fig. 8i) by ~35% from the nominal level, when reliability is at 0.7. It is this increase in pickup and delivery failure that results in the perception that reliability has less impact on latency at higher levels of congestion. It is the tasks and bundles that would have contributed most to a longer latency, that fail, such that their impact is not seen in the results. This relationship between latency and delivery ratio can lead to latency-focused missions wishing to discard large volumes of data, thus giving the perception of faster delivery times, so attention to the volumes delivered is critical.

If task and bundle lifetimes were extended, it is expected that the number of failed requests would decrease, but a further increase in latency would be realized as traffic congestion rises, slowing everything down. Interestingly, the difference in both the Request- and Pickup-ratios (Fig. 8d and e), from high to low reliability, remains relatively consistent across all congestion levels, suggesting that the *rate* of failure increases in a linear fashion as the number of accepted requests and issued tasks increases. Finally, Hop-count (Fig. 8f) undergoes an inflection at a RSL of ~0.6. Here, as congestion rises and reliability declines, an increasing number of bundles fail to traverse their preferred route and are either retained on-board until the next available download opportunity, or get dropped due to a lack of feasible downstream paths. Once dropped from the network, their hop-count is ignored from the statistic, resulting in a lower trend.

6. Conclusions

Contact Graph Scheduling (CGS) is a computationally lightweight option for satellite task scheduling, which could be used as the core mechanism for scheduling in satellite networks. It addresses a number of the shortcomings present with existing scheduling methods, including immediate processing of request arrivals, the consideration of how tasking information reaches the assigned satellite nodes, and a way to combine the pickup and delivery instructions into a single methodology. Many of these benefit stem from the combined scheduling-routing approach, such that consideration of paths taken by both tasking information and associated data is a core component of the algorithm.

CGS is capable of generating task allocations for delay-tolerant networks (DTN) with deterministic contact plans. The CGS concept provides a framework for converting requests for the pickup and delivery of goods, exploiting a Contact Graph representation of a time-varying graph, and traditional shortest-path algorithms. While the core objective of CGS is to minimize the time between request arrival to data product delivery, it intrinsically provides higher levels of request fulfillment than other approaches. In particular, resource awareness and the consideration of previously allocated tasks has a significant impact on network performance, specifically in the number of failed requests.

The impact of contact reliability is most clearly seen in the system's ability to fulfill requests through to data product delivery. This is due to the fact that contact uncertainty impacts both the timely pickup of data and the transfer of that data along its preferred route to destination, resulting in an increase to the number of data bundles that are dropped from the network due to expiry.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared details of the code/data within the manuscript

Acknowledgements

Funding: This work was supported by the Air Force Office of Scientific Research (AFOSR), award No. FA8655-22-1-7010, "Matryoshka Orbital Networks: empowering a shift from individual spacecraft to large networks of cooperative, self-organising agents".

References

[1] T. Abdelkader, K. Naik, N. Goel, V. Srivastava, A performance comparison of delay-tolerant network routing protocols, IEEE Netw. 30 (2016) 46–53, https://doi.org/10.1109/MNET.2016.7437024.
[2] M.A.N. Andini, Y. Satria, H. Burhan, Dynamic pickup and delivery problem with transfer in ridesharing to reduce congestion, in: Journal of Physics: Conference Series 1218, IOP Publishing, 2019, 012010, https://doi.org/10.1088/1742-6596/1218/1/012010.
[3] S. Augenstein, A. Estanislao, E. Guere, S. Blaes, Optimal scheduling of a constellation of earth-imaging satellites, for maximal data throughput and efficient human management, in: ICAPS 2016 (International Conference on Automated Planning Scheduling), London, UK, 2016, https://doi.org/10.1609/icaps.v26i1.13784.
[4] R. Bellman, On a routing problem, Q. Appl. Math. 16 (1958) 87–90.
[5] E. Bensana, G. Verfaillie, J.C. Agn`ese, N. Bataille, D. Blumstein, Exact and inexact methods for the daily management of an earth observation satellite. Space Mission Operations and Ground Data Systems 'SpaceOps '96, European Space Agency, 1996, p. 507.
[6] G. Berbeglia, J.F. Cordeau, G. Laporte, Dynamic pickup and delivery problems, Eur. J. Oper. Res. 202 (2010) 8–15, https://doi.org/10.1016/j.ejor.2009.04.024.
[7] N. Bezirgiannidis, C. Caini, V. Tsaoussidis, Analysis of contact graph routing enhancements for DTN space communications: analysis of CGR enhancements for DTN Space communications, Int. J. Satell. Commun. Network. 34 (2016) 695–709, https://doi.org/10.1002/sat.1138.
[8] E.W. Biefeld, in: PLAN-IT: Knowledge-Based Mission Sequencing, Cambridge, MA, 1987, p. 126, https://doi.org/10.1117/12.964866.
[9] E.J. Birrane, C. Caini, G.M. De Cola, F. Marchetti, L. Mazzuca, L. Persampieri, Opportunities and limits of moderate source routing in delay-/disruption-tolerant networking space networks, Int. J. Satell. Commun. Network. 40 (2022) 428–444, https://doi.org/10.1002/sat.1421.
[10] J. Boerkoel, J. Mason, D. Wang, S. Chien, A. Maillard, An efficient approach for scheduling imaging tasks across a fleet of satellites, in: International Workshop on Planning Scheduling For Space (IWPSS), 2021.
[11] P. Bouros, D. Sacharidis, T. Dalamagas, T. Sellis, Dynamic pickup and delivery with transfers. Advances in Spatial and Temporal Databases, Springer, Berlin, Heidelberg, 2011, pp. 112–129, https://doi.org/10.1007/978-3-642-22922-0-8.
[12] Burleigh, S., Fall, K., Birrane, E.J., 2022. Bundle protocol version 7. RFC 9171. URL: https://www.rfc-editor.org/info/rfc9171, doi:10.17487/RFC9171.
[13] C. Caini, G. De Cola, F. Marchetti, L. Mazzuca, Moderate source routing for DTN space networks, in: 2020 10th Advanced Satellite Multimedia Systems Conference and the 16th Signal Processing for Space Communications Workshop (ASMS/SPSC), Graz, Austria, IEEE, 2020, pp. 1–7, https://doi.org/10.1109/ASMS/SPSC48805.2020.9268926.
[14] C. Caini, G.M. De Cola, L. Persampieri, Schedule-aware bundle routing: analysis and enhancements, Int. J. Satell. Commun. Network. 39 (2021) 237–249, https://doi.org/10.1002/sat.1384.
[15] J.F. Cordeau, G. Laporte, The dial-a-ride problem: models and algorithms, Ann. Oper. Res. 153 (2007) 29–46, https://doi.org/10.1007/s10479-007-0170-8.
[16] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271, https://doi.org/10.1007/BF01386390.
[17] D. Eddy, M.J. Kochenderfer, A maximum independent set method for scheduling earth-observing satellite constellations, J. Spacecraft Rockets 58 (2021) 1416–1429, https://doi.org/10.2514/1.A34931.
[18] J.A. Fraire, O. De Jonck`ere, S.C. Burleigh, Routing in the space internet: a contact graph routing tutorial, J. Netw. Comput. Applic. 174 (2021), 102884, https://doi.org/10.1016/j.jnca.2020.102884.
[19] J.A. Fraire, E.L. Gasparini, Centralized and Decentralized routing solutions for present and future space information networks, IEEE Netw 35 (2021) 110–117, https://doi.org/10.1109/MNET.011.2100102.
[20] J.A. Fraire, P. Madoery, S. Burleigh, M. Feldmann, J. Finochietto, A. Charif, N. Zergainoh, R. Velazco, Assessing contact graph routing performance and reliability in distributed satellite constellations, J. Comput. Netw. Commun. 2017 (2017), https://doi.org/10.1155/2017/2830542 publisher: Hindawi.
[21] J.A. Fraire, P.G. Madoery, J.M. Finochietto, Traffic-aware contact plan design for disruption-tolerant space sensor networks, Ad. Hoc. Netw. 47 (2016) 41–52, https://doi.org/10.1016/j.adhoc.2016.04.007.
[22] N.G. Hall, M.J. Magazine, Maximizing the value of a space mission, Eur. J. Oper. Res. 78 (1994) 224–241, https://doi.org/10.1016/0377-2217(94)90385-9.
[23] F. Karami, W. Vancroonenburg, G. Vanden Berghe, A periodic optimization approach to dynamic pickup and delivery problems with time windows, J. Sched. 23 (2020) 711–731, https://doi.org/10.1007/s10951-020-00650-x.
[24] J. Kim, J. Ahn, H.L. Choi, D.H. Cho, Task Scheduling of agile satellites with transition time and stereoscopic imaging constraints, J. Aerospace Inf. Syst. 17 (2020) 285–293, https://doi.org/10.2514/1.I010775.

[25] H. Li, T. Zhang, Y. Zhang, K. Wang, J. Li, A maximum flow algorithm based on storage time aggregated graph for delay-tolerant networks, Ad. Hoc. Netw. 59 (2017) 63–70, https://doi.org/10.1016/j.adhoc.2017.01.006.
[26] Lowe, C., 2023. Contact graph scheduling Github code repository. https://github.com/chrisLoweDev/cgs.
[27] Lowe, C., 2023. Contact graph scheduling Python implementation. On zenodo.com. doi:10.5281/zenodo.7737582.
[28] Z. Lyu, A.J. Yu, The pickup and delivery problem with transshipments: critical review of two existing models and a new formulation, Eur. J. Oper. Res. 305 (2023) 260–270, https://doi.org/10.1016/j.ejor.2022.05.053.
[29] P.G. Madoery, J.A. Fraire, J.M. Finochietto, Congestion management techniques for disruption-tolerant satellite networks, Int. J. Satellite Commun. Network. 36 (2018) 165–178, https://doi.org/10.1002/sat.1210.
[30] S. Mitrovíc-Miníc, G. Laporte, The pickup and delivery problem with time windows and transshipment, INFOR: Inf. Syst. Oper. Res. 44 (2006) 217–227, https://doi.org/10.1080/03155986.2006.11732749.
[31] S. Nag, A.S. Li, J.H. Merrick, Scheduling algorithms for rapid imaging using agile Cubesat constellations, Adv. Space Res. 61 (2018) 891–913, https://doi.org/10.1016/j.asr.2017.11.010.
[32] Nag, S., Li, A.S., Ravindra, V., Net, M.S., Cheung, K.M., Lammers, R., Bledsoe, B., 2020. Autonomous scheduling of agile spacecraft constellations with delay tolerant networking for reactive imaging doi:10.48550/ARXIV.2010.09940.
[33] H.N. Psaraftis, A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem, Transport. Sci. 14 (1980) 130–154, https://doi.org/10.1287/trsc.14.2.130, publisher: INFORMS.
[34] A. Rais, F. Alvelos, M.S. Carvalho, New mixed integer-programming model for the pickup-and-delivery problem with transshipment, Eur. J. Oper. Res. 235 (2014) 530–539, https://doi.org/10.1016/j.ejor.2013.10.038.
[35] F.D. Raverta, J.A. Fraire, P.G. Madoery, R.A. Demasi, J.M. Finochi-etto, P. R. D'Argenio, Routing in delay-tolerant networks under uncertain contact plans, Ad. Hoc. Netw. 123 (2021), 102663, https://doi.org/10.1016/j.adhoc.2021.102663.
[36] C.A. Rigo, L.O. Seman, E. Camponogara, E. Morsch Filho, E.A. Bezerra, P. Munari, A branch-and-price algorithm for nanosatellite task scheduling to improve mission quality-of-service, Eur. J. Oper. Res. 303 (2022) 168–183, https://doi.org/10.1016/j.ejor.2022.02.040.
[37] M.W.P. Savelsbergh, M. Sol, The general pickup and delivery problem, Transport. Sci. 29 (1995) 17–29, https://doi.org/10.1287/trsc.29.1.17.
[38] A.P. Silva, K. Obraczka, S. Burleigh, J.M. Nogueira, C.M. Hirata, A congestion control framework for delay and disruption tolerant networks, Ad. Hoc. Netw. 91 (2019), 101880, https://doi.org/10.1016/j.adhoc.2019.101880.
[39] L. Torgerson, S.C. Burleigh, H. Weiss, A.J. Hooke, K. Fall, D.V.G. Cerf, K. Scott, R. C. Durst, Delay tolerant networking architecture, RFC 4838 (2007), https://doi.org/10.17487/RFC4838. URL, https://www.rfc-editor.org/info/rfc4838.
[40] A. Verma, Savita, S. Kumar, Routing protocols in delay tolerant networks: comparative and empirical analysis, Wirel. Personal Commun. 118 (2021) 551–574, https://doi.org/10.1007/s11277-020-08032-4.
[41] J.G. Walker, Some circular orbit patterns providing continuous whole earth coverage, Phys. Environ. Sci. (1970).
[42] M.J.H. Walker, B. Ireland, J. Owens, A set of modified equinoctial orbit elements, Celest. Mech. 36 (1985) 409–419, https://doi.org/10.1007/BF01227493.
[43] J. Wang, G. Song, Z. Liang, E. Demeulemeester, X. Hu, J. Liu, Unrelated parallel machine scheduling with multiple time windows: an application to earth observation satellite scheduling, Comput. Oper. Res. 149 (2023), 106010, https://doi.org/10.1016/j.cor.2022.106010.
[44] D. Wolfinger, J.J. Salazar-Gonźalez, The pickup and delivery problem with split loads and transshipments: a branch-and-cut solution approach, Eur. J. Oper. Res. 289 (2021) 470–484, https://doi.org/10.1016/j.ejor.2020.07.032.

Dr Christopher Lowe - Christopher is a Research Fellow in the Applied Space Technology Laboratory (ApSTL) within the Centre for Signal and Image processing (CeSIP). His research is focused on the design and exploitation of distributed space networks, such as constellations and federated satellite systems. Particular areas of interest are novel ways to operate satellite networks that feature intermittent connectivity, and routing of data through delay- and disruption-tolerant networks.

Dr Ruaridh Clark - Ruaridh is an engineer and network researcher who investigates influence and division in dynamical systems. His work often exists at the boundary between disciplines, and has led to publications on Alzheimer's disease, bird flocks, disease spread, image processing, robotic and satellite systems. This diverse portfolio informs his study of networks and offers opportunities to transfer innovation between domains.

Dr Ciara McGrath - Ciara McGrath is a Young Professional Member of AIAA and a Lecturer in Aerospace Systems at the University of Manchester, United Kingdom, with expertise in Astrodynamics and Space Mission Design. She is the Institution of Engineering and Technology's (IET) Young Woman Engineer of the Year 2021, and has appeared on TV, radio, and podcasts, and given a TEDx talk to promote engineering and the space industry to a wide audience. McGrath holds a Master's and Ph.D. in Aero-Mechanical Engineering from the University of Strathclyde, as well as a Postgraduate Certificate in Academic Practice.

Prof. Malcolm Macdonald - Malcolm Macdonald FRAeS FRSE FRSA is a Scottish space technology engineer, academic, and director. He is a Professor and the Chair of Applied Space Technology at the University of Strathclyde, and a visiting professor at University College Dublin. He was Director of the Scottish Centre of Excellence in Satellite Applications, SoXSA, from 2014 - 2020, and a non-executive member of the UK Space Agency Steering Board from 2017 - 2020. He is an acknowledged expert in space research, and in 2021 was referred to in the media as "Scotland's leading space expert".