Survivability in Lambda Grids by means of Ant Colony Optimization

Gustavo Sousa Pavani
Federal University of ABC (UFABC)
Santo André-SP, Brazil
gustavo.pavani@ufabc.edu.br

Andre Ricardo Frederico
Federal University of ABC (UFABC)
Santo André-SP, Brazil
andre.frederico@ufabc.edu.br

***Abstract*—Meta-scheduling in lambda grids is often a complex task because it typically comprises the discovery, monitoring, co-allocation, and orchestration of networking and computing resources. The support of advance reservations typically improves the performance of the lambda grid, but it also turns the meta-scheduling process much more complicated. All those mechanisms should deal with failures that may happen in the optical network or the computing infrastructure. Therefore, in this work, we propose a survivable, distributed grid meta-scheduler based on an Ant Colony Optimization (ACO) algorithm. By using restoration as the recovery mechanism, resilience against link, network node, and server node failure can be achieved. We evaluated the restorability for different combinations of meta- and local scheduling policies, and resource co-allocation algorithms under single link or single node failures. Besides, we assessed some of the parameters that may influence the restorability against server node failures, where the affected jobs are rescheduled to the remaining nodes of the grid. The results demonstrated that the ACO algorithm is capable of recovering near 100% of the jobs affected by link or server node failures for many of the combinations of meta- and local scheduling policies presented for the Server First-Relaxed (SF-R) and Network First (NF) co-allocation algorithms.**

***Index Terms*—Ant Colony Optimization; Restoration; Distributed meta-scheduling; Co-allocation of networking and computing resources; Advance reservations; Wavelength-routed optical networks.**

I. Introduction

An important requirement for communication infrastructures is to provide resilience against failures [1]. Indeed, communication links and network elements are subject to a myriad of unintentional and intentional failures [2], [3], which may disrupt the data flow and affect the users. Therefore, survivable communication infrastructure is necessary to overcome or mitigate the impact caused by failures, where survivability can be defined as the ability of a system to continue to provide service to its users in the presence of a failure event [1].

The goal of this paper is to present a survivable lambda grid framework based on the Ant Colony Optimization (ACO) metaheuristics [4]. Lambda grids [5], [6] refer to grids where their computing resources are interconnected by an optical network that allows its applications to dynamically request lightpaths on-demand.

This work was supported by São Paulo Research Foundation (FAPESP), grant no. 2015/24341-7.

978-3-903176-32-4

There are four important differences in the lambda grid compared to classical optical networks [7]: (i) anycast routing; (ii) best destination selection; (iii) advance reservation (AR) [6]; and (iv) job survivability against server node failures.

In effect, our previous work [8] covered the first three items, which are involved with the meta-scheduling process in the lambda grid control plane [5]. Meta-scheduling is the action of scheduling applications across different sites by orchestrating a pool of resources within each local scheduler. In fact, the distributed, self-organizing nature of ACO algorithms makes them promising candidates for (meta-)scheduling in traditional grids [9]–[13] as well as in lambda grids [8], [14].

Nevertheless, neither network survivability nor server node survivability in case of failures was studied in [8]. Thus, the contribution of this work is to extend the ACO algorithm [8] in order to allow the rescheduling of jobs in case of node failure [7], [15] in addition to lightpath survivability as already proposed in prior works regarding wavelength-routed optical networks [16]–[18]. Consequently, a new suitable destination has to be selected and the traffic re-routed accordingly while meeting the advance reservation constraints.

In this context, we propose different recovery mechanisms based on restoration for achieving both network and server node survivability in case of failures affecting the lambda grid. Indeed, up to our knowledge, few works with an ACO algorithm have proposed fault tolerance in job (meta-)scheduling for classical grids [9], [12] and none for lambda grids.

The remainder of the paper is organized as follows. In Section II, we present the lambda grid architecture, detailing the meta-scheduling and the advance reservation model used throughout this work. We present the Ant Colony Optimization algorithm used in this work. We propose the survivability mechanisms for lambda grids in Section IV, detailing the simulations carried out for assessing them in Section V. The results obtained through simulations are shown and discussed in Section VI. Finally, in Section VII, conclusions are drawn.

II. Architectural Model

The lambda grid control plane provides the discovery, monitoring, co-allocation, and provisioning of computing and networking resources. Hence, the control plane presents a uniform interface to lambda grid applications to make a dynamic, orchestrated use of the grid resources.

In this work, we assume a hybrid GMPLS [19] control plane for the lambda grid with an integrated model similar to [20]. In this context, we can identify a grid layer, which comprises the lambda grid applications and computing resources, and a network layer, which comprises the networking resources.

The availability of resources of both layers is seamlessly gathered by the artificial ants, whereas the resources of both layers are reserved and allocated by a single signaling protocol. Thus, the artificial ants are responsible for the discovery of resources and their respective monitoring, whereas the signaling protocol is responsible for the reservation and actual allocation of the resources.

Indeed, artificial ants work as control messages exchanged by the lambda grid nodes at the control plane, replacing the routing protocol and the grid middleware while maintaining the signaling protocol essentially unaltered [17]. As a result, artificial ants co-exist with signaling messages in the control channels of the lambda grid control plane.

A. Meta-scheduling and advance reservation

In this work, we consider a distributed meta-scheduler [8], [21]. In this model, each resource in the grid has a local scheduler and the meta-scheduler decides which local scheduler it will delegate the job. The local scheduler is commonly referred to as the Local Resource Management System (LRMS) and it independently decides how to locally assign the job to its resources. We assume a single LRMS instance per each grid site.

In this case, although each LRMS has detailed information about the availability of its processing and networking resources, the distributed meta-scheduler has neither total control nor full knowledge of each site's resources. We also assume that advance reservations are supported [6], [8], where the use of resources can be delayed until a future time.

By using the resource information gathered by the ants, three different heuristics can be employed by the meta-scheduler to select the best destination site [8]: Least-Loaded (LL) policy, Closest Available (CA) policy, and Best Availability-Distance Ratio (BADR) policy.

Then the meta-scheduler can send a signaling message to the chosen destination site. Upon the reception of the message, the LRMS site instance has to locally schedule a job to a resource within the horizon time $T_{horizon}$, which indicates the time range from the current time $T_{current}$ to the latest available time that it is possible to schedule a reservation. We assume the following local resource scheduling policies [8]: First-Fit (FF) and Earliest Starting Time (EST).

We assume that an advance reservation can specify a range of starting times and the duration. This approach is referred to as Specified starting Time, Specified Duration (STSD) with flexible window [6], [22] or first-fit/deadline [23]. The actual starting time for a reservation T_{start} belongs to the interval $[T_{earliest}; T_{latest}]$, where $T_{earliest}$ is the earliest available starting time and T_{latest} is the latest available starting time for beginning a reservation. $T_{window}^{interval}$ specifies the flexible window duration, i.e., the time interval allowed for a reservation.

For the request times in the flexible time window $[T_{earliest}; T_{latest}]$, we consider [8]:

$$T_{earliest} = T_{current} + T_{window}^{min} + (T_{window}^{max} - T_{window}^{min} - T_{window}^{interval})U(1) \quad (1)$$

$$T_{latest} = T_{earliest} + T_{window}^{interval} \quad (2)$$

where $T_{current}$ is the current time, T_{window}^{min} is the minimum start time of the flexible window, T_{window}^{max} is the maximum start time of the flexible window, $T_{window}^{interval}$ is the flexible window time interval and $U(1)$ is a uniform random number generator $\in [0; 1]$. Note that $T_{earliest} \in [T_{current} + T_{window}^{min}; T_{current} + (T_{window}^{max} - T_{window}^{interval})]$ and $(T_{window}^{max} - T_{window}^{min} - T_{window}^{interval}) > 0$.

We assume that both networking duration D_{net} and processing duration D_{proc} are known in advance. The sum of those durations determines the whole duration of the job. D_{net} can be decomposed into transmission time, i.e., how long a lightpath is active to transport the data related to the job, and guard time T_{guard}, which is the interval between two consecutive lightpaths on the same channel [8].

Figure 1 illustrates the advance reservation model for requests used throughout this work. The book-ahead time $T_{book-ahead}$ is the time difference between $T_{earliest}$ and $T_{current}$.

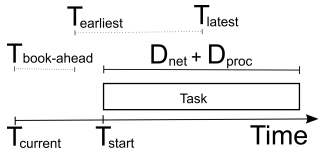


Fig. 1. The advance reservation model.

III. Ant Colony Optimization (ACO)

The class of algorithms known as Ant Colony Optimization is inspired by the foraging behavior of natural ants. Natural ants lay down chemical markers, which are called pheromones, to develop trails connecting the nest to different food sources. This type of indirect communication based on modifications of the environment to stimulate subsequent actions is referred to as stigmergy [24]. It allows the ant colony to exhibit an emergent and self-organizing behavior even though the ants are very simple, lightweight agents.

AntNet [25] is an algorithm based on ACO metaheuristics whose main application is routing in telecommunication networks. Indeed, it has proved to be an effective approach for routing packets in packet-switching networks [25], being competitive with other state-of-the-art algorithms.

The original AntNet algorithm was later adapted to wavelength routing [17], IP-over-Optical networks [18], [26], and Optical Packet Switching (OPS) technology [27]. Instead of using the delay introduced at each hop as the metric for routing as in [25], these adaptations consider the number of traversed hops, which are stored in the ants' memory. Thus, each ant carries the identification of each node it traverses.

In the lambda grid environment, the role of the artificial ants also includes the collection of the processing availability

information at their destination sites for advance reservation purposes. This additional information gathered by the ants allows for grid resource discovery problem to be naturally combined with routing, i.e., the discovery of an appropriate route to connect this resource to a grid application. In this context, AntNet can be seen as a viable solution for distributed meta-scheduling and orchestration in lambda grids [8], [14].

A. Discovery and monitoring of computing and networking resources

At regular intervals ($1/L_{ants}$), an ant is sent from a random source node s to a random destination node d. This ant is called forward ant and it gathers the label of each node i where it traverses, putting it in its memory $V_{s \to i}$, which also serves as a tabu list. The forward ant selects the next hop among the set of neighbor nodes by using a probabilistic rule that accounts for the pheromone levels in each neighbor link and local information based on wavelength availability.

When a forward ant arrives at its destination d, it becomes a backward ant. It probes the LRMS for processing resources. The future availability information has to be summarized to reduce the communication overhead and to provide confidentiality [28]. In this work, a time-slotted approach was employed to aggregate processing consumption information at each site [29]. Hence, the timeline is broken into a set of timeslots. For sake of simplicity and easiness of implementation, we consider static timeslots, which have a constant length in the time-domain ($T_{timeslot}$), and the same horizon length ($T_{horizon}$) as shown for the local scheduling.

Thus, at the destination node, the backward ant gathers the processing availability information for a random static timeslot within the horizon length by assessing the number of idle processors during the whole duration of that timeslot. Next, it returns to the source node using the same path followed when it was a forward ant. In its trip back to the source node and along with the traversed nodes, the backward ant will update the statistical parametric model, by using an exponential model [17], [25], and the pheromone routing table, based on the goodness of the route found [17], [25].

When the backward ant returns to source node s, it will update the correspondent timeslot and site destination in the Future Availability Table (FAT) [8]. The FAT has an entry to store the processing availability for each destination site.

The information stored in the FAT regarding all stored timeslots is used by the local meta-scheduler to select the best destination site, as shown in Subsection II-A, when a reservation request is issued by a local lambda grid application. Note that each grid node has its own local, independent meta-scheduler since the FAT only stores the future availability information collected by the ants generated by that node. Thus, each grid node can effectively manage the resource discovery and monitoring based on its own information about the lambda grid. Consequently, there is no single point of failure for the resource discovery and monitoring mechanism in the lambda grid [15].

In summary, the artificial ants are continuously probing the state of the lambda grid and updating their local data structures, which can be later used for best destination selection and routing[1]. In effect, when the destination site is selected accordingly to the chosen policy, our anycast problem is reduced to an advance reservation co-allocation problem of computing and networking resources.

B. Co-allocation and provisioning of computing and networking resources

The actual reservation and allocation of the computing and processing resources are done by the RSVP-TE signaling protocol [30]. Thus, when an RSVP-TE signaling message arrives at an LRMS, the set of already accepted reservations is used to verify if that request can be fulfilled by available resources at the site as shown in Subsection II-A. This approach is known as reservation-based [6], [29].

In AntNet, the process of finding good routes in the network is fully distributed. We assume that all nodes can obtain crankback information to re-route lightpath setups based on current values of the pheromone-routing table [17], [31], [32]. The crankback mechanism allows for the search of alternate, feasible routes when establishing a lightpath between a resource and a grid application. Hence, it improves the solution of the Routing and Wavelength Assignment (RWA) problem with advance reservation demands [8], in which a first-fit approach is used as the wavelength assignment algorithm.

We also assume that the signaling protocol can be extended to support advance reservations on both networking and processing resources by including new RSVP-TE objects [8]: *Label Resource Request* (LRR) and *Advance Label Set* (ALS).

Three distributed algorithms for solving the joint computing and networking resource advance allocation and reservation problem in lambda grids were proposed in [8]: Server First (SF), Server First-Relaxed (SF-R), and Network First (NF), in which a detailed explanation of the exchange of RSVP-TE messages can be found.

IV. Survivability

In a lambda grid system, unplanned failures may disrupt the communication of the underlying optical network or take down the computing resources at a node/site. Therefore, a survivable lambda grid is sought to recover from the impact of failures on its operation.

One of the most common failures in optical networks is fiber optical cable cuts [1]. They interrupt the communication on the links of the network. The situation in which the link between two adjacent optical network nodes fails is called a single-link failure.

Equipment failures are also common in optical networks. They disrupt the operation of the affected node. The situation in which a network node fails is called a single-node failure. Since a node failure puts all of its attached links out of service, it can be seen as a special case of multiple link failures. In

[1]Please refer to [8], [17], [25], [26] for a more detailed explanation of the AntNet algorithm.

both cases, the network layer at the control plane can react very quickly when a failure occurs.

On the contrary, computational resources on the lambda grid affected by a node failure may not be subject to a fast recovery. Since the grid meta-scheduling is managed by ants, which sample the grid at a lower rate and are prone to be lost, it could take a long time before the grid layer realizes that a node is down. Note that timeslot discretization also makes the meta-scheduler slower to react to failures.

Moreover, a node failure can cause a massive number of jobs trying to be restored at the same time. Out-of-date information at the FAT of the source nodes may cause many jobs to be redirected to already fully-loaded nodes, which cannot handle the jobs in excess. Thus, we considered the following coordinated escalation scheme for recovering jobs on the lambda grid. First, the network layer tries to recover from the failure, because it can react much faster than the grid layer. Then, only in the case of a node failure, the recovery process is escalated to the grid meta-scheduler of each source node, which will try to find a new destination to reallocate their affected jobs.

In this work, we consider restoration, which uses the available spare capacity to restore the system after the occurrence of the failure, as the recovery mechanism. In effect, restoration is typically more resource-efficient than protection [15], where resources are pre-reserved before the occurrence of a failure. However, restoration techniques cannot guarantee survivability.

In order to assess the resilience of the lambda grid in failure scenarios, we define the restorability R metric:

$$R = \frac{J_{restored}^{failed}}{J^{failed}} \tag{3}$$

where $J_{restored}^{failed}$ is the number of jobs affected by the failure(s) and restored by the recovery process and J^{failed} is the total number of jobs affected by the failure(s).

A. Network layer recovery

For network recovery, we consider the mechanism presented in [17], which employs end-to-end restoration without route pre-computation [33]. We also consider that the resources of a lightpath affected by failure are released before the establishment of an alternate lightpath and the time I_{detect} is needed to detect and localize a network failure.

Indeed, ACO-based routing can take advantage of local crankback information to recover from a network failure [17] even in the presence of inaccurate, out-of-date information at the pheromone routing tables [32].

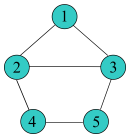
During this recovery process, we consider that the computational resources associated with an active or scheduled lightpath affected by failure are released by the local LRMS only if the restoration process cannot restore the lightpath. In other words, the job retains the computational resources originally allocated to it during the restoration process and only the networking reservation is re-scheduled. This scheme is called make-before-break and it is also known as soft restoration [34].

We assume that a job whose active or scheduled lightpath could not be restored is considered blocked. It is worth mentioning that, in the case of single-node failure, the network restoration mechanism may recover an active or scheduled lightpath only if the affected node is not its destination. Besides, the optical network is not aware of the jobs executing at the grid nodes. Hence jobs assigned to a failed node can only be recovered in the next phase, i.e., when the meta-scheduler is escalated to try to restore them.

B. Grid layer recovery

In order to detect and locate the failures on the server nodes, a watchdog timer is associated with the FAT.

A node d in the FAT will be considered failed if there is no arrival of a backward ant with destination to d within the interval $I_{watchdog}$. So, the watchdog timer will notify the meta-scheduler about the failure of node d. In general, $I_{watchdog} \gg I_{detect}$. Figure 2 depicts an example of the watchdog timer with $I_{watchdog}$ = 20 s associated with a Future Availability Table with a fixed timeslot equal to 100 s and a horizon length equal to 500 s. Note that all destinations nodes in the example are considered to be functioning as a backward ant arrived less than 20 seconds ago.



Destinations	Watchdog Interval	Slot Times 100	200	300	400	500
2	10	35	37	40	41	42
3	4	20	28	28	30	31
4	7	70	76	79	80	84
5	2	50	56	56	60	62

Fig. 2. Example of a FAT with a watchdog timer for node 1.

Thus, in case of the expiration of the watchdog timer for a destination node d, the meta-scheduler will remove it from the FAT and allocate its unfinished jobs to another grid node.

At regular intervals I_{jobs}, the meta-scheduler will choose a single job to be restored and it will select a new destination site according to its meta-scheduling policy. Trying to recover one job at a time permits the FAT to be updated by the ants while avoiding the overload of a single destination site. Let us consider that, at the moment of the job recovery, we have time $T_{current}$. The new value of $T_{earliest}$ will be:

$$T_{earliest} = T_{current} + \Delta \tag{4}$$

where Δ is a parameter that delays the minimum starting time of a reservation and $\Delta \in [T_{window}^{min}; (T_{window}^{max} - T_{window}^{interval})]$. Note that a higher value of Δ increases the book-ahead time and, consequently, the chances to reschedule the job.

Since the grid layer is responsible for the recovery of server node failures, the restorability for these failures is also referred to as grid restorability in the remainder of the paper.

V. Simulations

For evaluating the proposed algorithms, we used the NSFNet backbone network [8], [25], which is a 14-node network with 21 bidirectional links.

We consider a highly asymmetric scenario, where nodes 1 and 2 are source nodes that host grid applications and compete with the same load for processing resources located in the remaining nodes of the network. The source nodes are the only ones that generate ants [14]. Thus, nodes 1 and 2 will simulate two independent meta-schedulers that will contend for resources located elsewhere in the lambda grid, being individually informed about failures by the grid control plane.

We consider all possible link failures, while node failures are simulated for nodes 3 to 14 only. All failures considered four different times of simulation.

The duration of each lightpath follows an exponential distribution with a mean value of 10 s and we assume there is no blocking when the results of the job execution are sent back to the source node. We consider 4 wavelengths available on each network link.

We consider a homogeneous pool of processing resources at each site equal to P_{proc} processors, where a job gets only one processor. The execution time of each job follows a negative exponential service time with an average value of 1500 s.

Some of the parameters used in the simulations are depicted in Table I. The remaining simulation parameters used in the simulations are the same as shown in [8], [17].

TABLE I
PARAMETERS USED IN THE SIMULATIONS.

Parameter	Symbol	Value
Global rate for launching forward ants	L_{ants}	48 ants/s
Total number of processors at each node	P_{proc}	100
Minimum start time of the window	T^{min}_{window}	2 s
Maximum start time of the window	T^{max}_{window}	2000 s
Flexible window time interval	$T^{interval}_{window}$	1850 s
Horizon time for (meta-)scheduling	$T_{horizon}$	2000 s
Guard time between lightpaths	T_{guard}	10^{-3} s
Interval to detect and isolate the failure	I_{detect}	10^{-2} s
Timeout for the watchdog timer	$I_{watchdog}$	20 s

VI. RESULTS

We considered the following notation on the next figures, which takes the form: meta-scheduling policy (CA, LL, or BADR) – local scheduling policy for the processing resources (EST or FF) / local scheduling policy for the networking resources (EST or FF). Since the EST and FF local scheduling policies are equivalent to the networking resources [8] when the SF algorithm is used, it is omitted for sake of clarity. The error bars indicate the standard error of the mean.

In Figure 3, the network restorability under single link failure is evaluated. For almost all grid workloads, we have restorability over 80% for the SF algorithm. For the SF-R algorithm, we can observe a restorability over 95% for all but the LL–EST/EST and LL–EST/FF combinations. We can also observe that the restorability is improved compared to the SF algorithm. For the NF algorithm, except for the local scheduling combination of EST/EST, we have network restorability above 95% and we can observe a small improvement in the restorability compared to the SF-R algorithm.

Figure 4 depicts the network restorability under node failure. The behavior of the SF algorithm under node failure regarding the combinations is similar to the behavior under link failure as seen in Figure 3a. However, we can observe that the restorability is much smaller, ranging from 25% to 50%. Again, for the SF-R algorithm, the performance trend regarding the combinations is similar to the link failure scenario as seen in Figure 3b. As already noted for the SF algorithm, the restorability is much smaller under single node failure compared to the link failure scenario, ranging from 20% to 55%. The restorability for the NF algorithm ranges from 15% to 55%, being quite similar to the restorability shown for the SF-R algorithm.

In effect, as a node failure may affect multiple links, there may be not enough spare capacity at the underlying optical network to restore the lightpaths that traversed the failed node.

As already mentioned, the jobs assigned to a failed node can only be recovered by the grid layer. However, the grid layer may have a much slower convergence than the network layer. For this reason, we first evaluate two parameters that may affect the restorability at the grid layer as discussed in Subsection IV-B: the interval I_{jobs} and the Δ time.

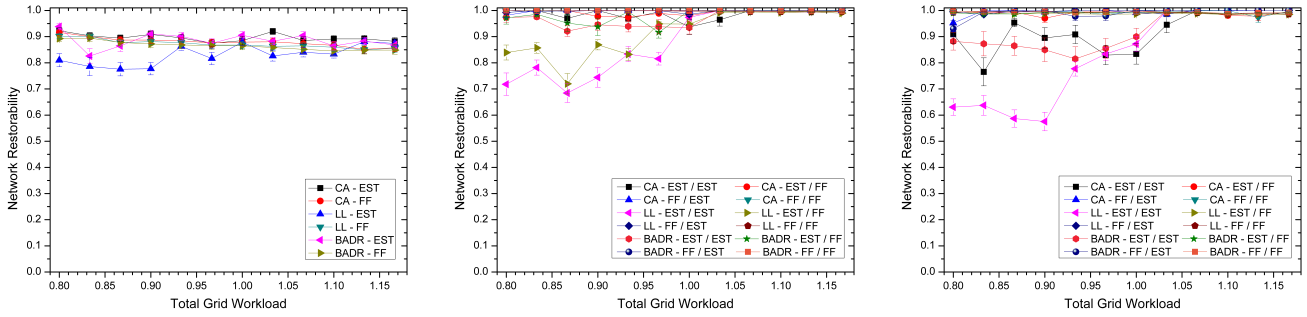
Hence, the results for different values of I_{jobs} and Δ at a 100% total grid workload are shown in Figure 6. We consider the best performing combination in terms of blocking probability for each algorithm as indicated in [8].

We can observe that the NF combination is not affected by either the values of I_{jobs} or Δ. On the contrary, the SF-R combination is greatly impacted by both the I_{jobs} and Δ values. The SF combination is more affected by the value of Δ than the value of I_{jobs}. As the values of I_{jobs} and Δ increases, the grid restorability is improved as we have more time to update the FAT and a larger book-ahead time, respectively.

For I_{jobs} = 8 seconds and Δ = 150 seconds, which is the maximum allowable value, the probability for the SF and SF-R algorithms to recover a job becomes very high ($>$ 95%) and it almost reaches the same restorability value for the NF algorithm. Thus, these values are assumed for the remaining simulations of this work.

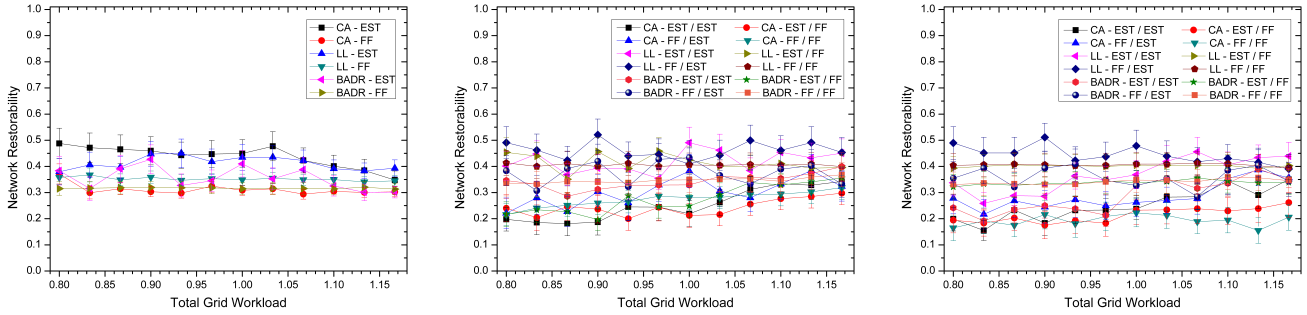
Finally, Figure 5 depicts the recovery of the jobs affected by a node failure after the correspondent network recovery.

For the SF algorithm, the LL–EST combination achieves the best grid restorability, whereas the worst-performing combination is CA–FF. The grid restorability for all combinations ranges from 80% to 100%. For the SF-R algorithm, we can note an improvement over the SF algorithm, with the grid restorability ranging from 90% to 100%. The worst performing combinations are the CA policy with FF scheduling for the local resources. Indeed, looking for the closest available node to restore a failed job is not a good strategy as it can be saturated by the occurrence of a failed node. In other words, it is a better strategy to try to accommodate the failed jobs more evenly across the lambda grid. For the NF algorithm, the CA–FF/FF followed by the CA–EST/FF and CA–FF/EST are the worst-performing combinations, with grid restorability ranging from 75% to 95%. The remaining combinations have grid restorability values equal to or greater than 95% until 100% total grid workload.



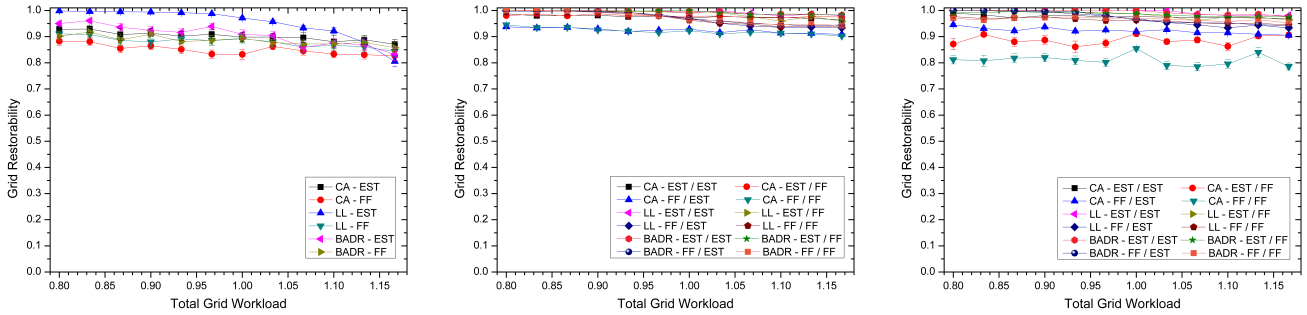
(a) SF algorithm. (b) SF-R algorithm. (c) NF algorithm.

Fig. 3. Network restorability under link failure.



(a) SF algorithm. (b) SF-R algorithm. (c) NF algorithm.

Fig. 4. Network restorability under node failure.



(a) SF algorithm. (b) SF-R algorithm. (c) NF algorithm.

Fig. 5. Grid restorability.

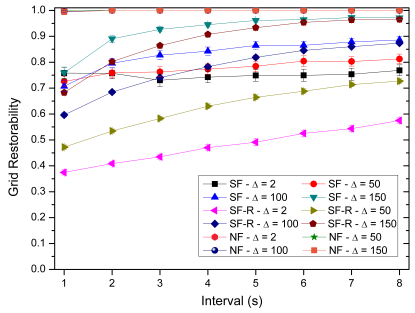


Fig. 6. Grid restorability for different intervals and Δs.

VII. Conclusions

In this work, we presented a survivable, distributed meta-scheduler based on ACO metaheuristics with advance reservation support in lambda grids. We proposed recovery mechanisms for achieving resilience against network and node server failures. A classical restoration mechanism was used for the underlying optical network. Besides, due to the distributed nature of the meta-scheduling process, a watchdog timer was incorporated into the FAT to allow for the detection and restoration of the server node failures, where affected jobs are rescheduled to non-failed nodes.

We evaluated the restorability under different meta- and local scheduling policies against single link/node failures. The proposed algorithm was capable of recovering near 100% of the jobs affected by link or server node failure for many of the combinations of meta- and local scheduling policies presented for the SF-R and NF co-allocation algorithms. We also have shown that SF and SF-R algorithms may have the grid restorability largely impacted by the time needed to update the FAT and the book-ahead time used for job restoration.

REFERENCES

[1] R. Ramaswami, K. Sivarajan, and G. Sasaki, *Optical Networks: A Practical Perspective*, 3rd ed. Morgan Kaufmann, 2009.

[2] J. Vasseur, M. Pickavet, and P. Demeester, *Network recovery: protection and restoration of optical, SONET-SDH, IP, and MPLS*. Morgan Kaufmann, 2004.

[3] W. D. Grover, *Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*. Prentice Hall, 2003.

[4] M. Dorigo and T. Stützle, *Ant Colony Optimization*. MIT Press, 2004.

[5] F. Palmieri, "Network-aware scheduling for real-time execution support in data-intensive optical grids," *Future Generation Computer Systems*, vol. 25, no. 7, pp. 794–803, July 2009.

[6] N. Charbonneau and V. M. Vokkarane, "A survey of advance reservation routing and wavelength assignment in wavelength-routed WDM networks," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1037–1064, Fourth Quarter 2012.

[7] A. Shaikh and B. Jaumard, "Optimized dimensioning of resilient optical grids with respect to grade of services," *Telecommunication Systems*, vol. 56, pp. 189–200, 2014.

[8] G. S. Pavani and R. I. Tinini, "Distributed meta-scheduling in lambda grids by means of ant colony optimization," *Future Generation Computer Systems*, vol. 63, pp. 15–24, October 2016.

[9] S. Guo, H.-Z. Huang, Z. Wang, and M. Xie, "Grid service reliability modeling and optimal task scheduling considering fault recovery," *IEEE Transactions on Reliability*, vol. 60, no. 1, pp. 262–274, March 2011.

[10] E. Pacini, C. Mateos, and C. G. Garino, "Distributed job scheduling based on swarm intelligence: A survey," *Computers and Electrical Engineering*, vol. 40, no. 1, pp. 252–269, January 2014.

[11] P. K. Tiwari and D. P. Vidyarthi, "Improved auto control ant colony optimization using lazy ant approach for grid scheduling problem," *Future Generation Computer Systems*, vol. 60, pp. 78–89, July 2016.

[12] H. Idris, A. E. Ezugwu, S. B. Junaidu, and A. O. Adewumi, "An improved ant colony optimization algorithm with fault tolerance for job scheduling in grid computing systems," *PLoS ONE*, vol. 12, no. 5, pp. 1–24, May 2017.

[13] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Generation Computer Systems*, vol. 91, pp. 407–415, 2019.

[14] G. S. Pavani and H. Waldman, "Co-scheduling in lambda grid systems by means of ant colony optimization," *Future Generation Computer Systems*, vol. 25, no. 3, pp. 257–265, March 2009.

[15] Y. Chen, A. Bari, and A. Jaekel, "Strategies for fault tolerance in optical grid networks," in *International Conference on Computing, Networking and Communications (ICNC 2012)*, 2012, pp. 628–632.

[16] G. S. Pavani and H. Waldman, "Restoration in wavelength-routed optical networks by means of ant colony optimization," *Photonic Network Communications*, vol. 16, no. 1, pp. 83–91, August 2008.

[17] ——, "Routing and wavelength assignment with crankback re-routing extensions by means of ant colony optimization," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 4, pp. 532–541, May 2010.

[18] K. S. Amorim and G. S. Pavani, "Routing and restoration in IP/MPLS over optical networks by means of ant colony optimization," in *IEEE Global Communications Conference (GLOBECOM 2019)*, 2019.

[19] E. Mannie, "Generalized multi-protocol label switching (GMPLS) architecture," RFC 3945 (Proposed Standard), Internet Engineering Task Force, Oct. 2004.

[20] N. Ciulli, G. Carrozzo *et al.*, "Architectural approaches for the integration of the service plane and control plane in optical networks," *Optical Switching and Networking*, vol. 5, no. 2, pp. 94–106, June 2008.

[21] S. Sotiriadis, N. Bessis, F. Xhafa, and N. Antonopoulos, "From meta-computing to interoperable infrastructures: A review of meta-schedulers for HPC, grid and cloud," in *IEEE International Conference on Advanced Information Networking and Applications (AINA 2012)*, 2012, pp. 874–883.

[22] J. Zheng and H. T. Mouftah, "Routing and wavelength assignment for advance reservation in wavelength-routed WDM optical networks," in *IEEE International Conference on Communications (ICC 2002)*, 2002, pp. 2722–2726.

[23] L.-O. Burchard, "Advance reservations of bandwidth in computer networks," Ph.D. dissertation, Technischen Universität Berlin, 2004.

[24] P. P. Grassé, "La reconstruction du nid et les coordinations interindividuelles chez Bellicoitermes natalenis et Cubitermes sp. La théorie de la stigmergie: Essai d'interprétation des termites constructeurs," *Insectes Sociaux*, vol. 6, pp. 41–81, 1959.

[25] G. Di Caro and M. Dorigo, "AntNet: distributed stigmergetic control for communications networks," *Journal of Artificial Intelligence Research*, vol. 9, pp. 317–365, 1998.

[26] K. S. Amorim and G. S. Pavani, "Ant colony optimization-based distributed multilayer routing and restoration in IP/MPLS over optical networks," *Computer Networks*, vol. 185, p. 107747, 2021.

[27] G. S. Pavani and H. Waldman, "Traffic engineering and restoration in optical packet switching networks by means of ant colony optimization," in *International Conference on Broadband Communications, Network and Systems (BroadNets 2006)*, 2006, pp. 1–9.

[28] P. Kokkinos and E. A. Varvarigos, "Scheduling efficiency of resource information aggregation in grid networks," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 9–23, January 2012.

[29] C. Barz, U. Bornhauser, P. Martini, and M. Pilz, "Timeslot-based resource management in grid environments," in *IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN 2008)*, 2008, pp. 39–48.

[30] L. Berger, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions," RFC 3473 (Proposed Standard), Jan. 2003.

[31] A. Farrel, A. Satyanarayana, A. Iwata, N. Fujita, and G. Ash, "Crankback signaling extensions for MPLS and GMPLS RSVP-TE," RFC 4920 (Proposed Standard), Internet Engineering Task Force, Jul. 2007.

[32] G. S. Pavani, A. F. Queiroz, and J. C. Pellegrini, "Analysis of ant colony optimization-based routing in optical networks in the presence of byzantine failures," *Information Sciences*, vol. 340–341, pp. 27–40, 2016.

[33] D. Papadimitriou and E. Mannie, "Analysis of Generalized Multi-Protocol Label Switching (GMPLS)-based Recovery Mechanisms (including Protection and Restoration)," RFC 4428 (Informational), Internet Engineering Task Force, Mar. 2006.

[34] E. Mannie and D. Papadimitriou, "Recovery (Protection and Restoration) Terminology for Generalized Multi-Protocol Label Switching (GMPLS)," RFC 4427 (Informational), Internet Engineering Task Force, Mar. 2006.