An Efficient Scheduling Algorithm for Time-Driven Switching Networks

Thu-Huong Truong♦ , Mario Baldi*, Yoram Ofek♦
♦Department of Information and Communication Technology
University of Trento, Italy
Email: (huong.truong, ofek)@dit.unitn.it
*Control and Computer Engineering Department
Politechnico di Torino, Italy
Email: mario.baldi@polito.it

Abstract— **Time-driven Switching (TDS) networks with non-immediate forwarding (NIF) provides scheduling flexibility and consequently, reduces the blocking probability (blocking is defined to take place when transmission capacity is available, but without a feasible schedule). However, it has been shown that with NIF scheduling complexity may grow exponentially. Efficiently finding a schedule from an exponential set of potential schedules is the focus of this paper. The work first presents the mathematical formulation of the NIF scheduling problem, under a wide variety of networking requirements, then introduces an efficient (i.e., having at most polynomial complexity) search algorithm that guarantees to find at least one schedule whenever such a schedule exists. The novel algorithm uses 'trellis' representations and the well-known survivor-based searching principle.**

Index Terms— **scheduling, search algorithms, time-driven switching, pipeline forwarding, optical networks**

I. INTRODUCTION

Scheduling for flexible bandwidth provisioning in heterogeneous networks while satisfying various service requirements is critical in next generation networking. The main context of this work is time-driven Switching (TDS), see [1]-[6], which is a scalable switching design based on UTC (Coordinated universal time) with pipeline forwarding. Under the pipeline forwarding principle packets are forwarded in time frames (TFs) in a "lock-step" manner across the route. TDS enables deterministic performance guarantees, flexible bandwidth provisioning, and low cost switching scalability.

Pipeline forwarding at a TDS switch can be performed in two manners (1) immediate forwarding (IF) and (2) non-immediate forwarding (NIF). IF is simple but provides a smaller number of different pipeline forwarding schedules, and consequently, may result in high blocking probability (blocking is defined as an event in which transmission capacity is available without a feasible schedule). On the other hand, NIF provides higher scheduling flexibility as the number of possible schedules growing exponentially with the number of hops, and consequently, significantly reducing the blocking probability. The complexity of TDS scheduling problem depends on various factors, such as, the forwarding schemes (IF, NIF), the network dimension (the number of switches, the number of wavelengths per optical fiber), the predefined technology parameters (link bandwidth, the duration of time frames and time cycles).

The schedule search algorithm presented in [2] is suitable only for the simple IF case of single channel per link, not dealing with the complexity introduced by WDM and NIF, which is the focus of this paper. The work [7] addresses the RWTA (Route, Wavelength, Time slot Assignment) problem in time-shared wavelength-routed WDM networks. Although this has similarities with the scheduling task in TDS networks, [7] only deals with a scenario that is comparable to the IF case. Scheduling a scenario featuring IF and no wavelength conversion has lower complexity (time slot and wavelength assignment) but less scheduling/provision flexibility.

Within the scope of this paper, we will present an efficient algorithm for the NIF problem of time frame scheduling over a predefined route with extensions to multiple-wavelength. The paper is organized as follows: Section II formulates our problem and shows the way that led to our proposed solution. In Section III, we first present an algorithm for the fundamental case of single-TF request in a single-channel, homogeneous network (all links have the same capacity). A special graph, i.e. a trellis, is constructed and used by the per-request search algorithm that is motivated by the Viterbi algorithm [10] and compared with the well known Dijkstra algorithm [11][12]. Section IV extends the solution to the more complicated case of WDM homogeneous networks. Finally, we discuss extensions of this work in Section V.

II. SCHEDULING PROBLEM FORMULATION AND SCOPE

This work focuses on a time-driven switching (TDS) network with an arbitrary topology, where each optical link transports one or more optical channels (lambdas) with defined transmission bit rates. The TDS network operation principles were described in depth in [2]. The following is a brief summary that is needed for understanding of our scheduling search design and analysis.

A. TDS Network principle

Time Structure: TDS network uses common time reference (CTR) that is commonly realized by using *UTC* (coordinated universal time). UTC is available everywhere through GPS and Galileo in the near future with accuracy that is well below 1μs. As Figure 1 depicts, one standard UTC second is divided into equal duration time frames (TFs), which are grouped into time cycles (TCs), such that, multiple contiguous TCs are equal to one UTC second. TFs are used to align and pipeline-forward multi-protocol packets between switches. The TF capacity is calculated according to its duration and the link bandwidth. In our assumption there are K TFs per TC and all links are having the same TC duration.

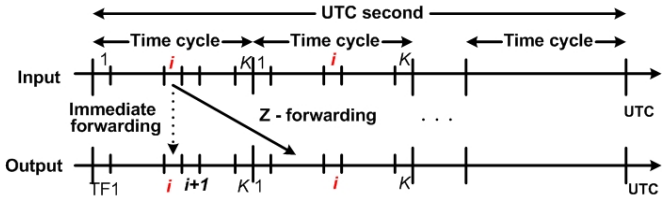

Figure 1- Time structure and pipeline forwarding

Pipeline Forwarding: The basic principle of TDS network operation is pipeline forwarding (PF), in which packets are forwarded in TFs with a predefined forwarding schedule that is responsive to UTC and without header processing. Consequently, TFs can be viewed as virtual containers of packets. The necessary condition for pipeline forwarding is having delay between inputs of TDS switches to be an integer number of TFs. In order to realize this all incoming TFs should be aligned with UTC. **However, without loss of generality, in this work we presume the availability of this alignment operation and ignore the propagation delay.**

Pipeline forwarding delay is the delay, measured in TFs, of one hop between the inputs of two neighboring switches on a route. In fact, the forwarding delay comprises of the propagation delay and the necessary UTC alignment delay (which we assume to be zero in the following analysis) and the Z-forwarding delay, which is the scheduling delay due to holding the incoming TF (with its packets) for a duration between 0 and Z TFs before forwarding to the next TDS switch on the route.

The Z-forwarding has two basic cases, as shown in Figure 1:

1. $Z=0$ – Immediate Forwarding (IF): incoming TFs are forwarded with zero delay to the next switch.
2. $K>Z>0$ – Non-Immediate Forwarding (NIF): incoming TFs can be forwarded to the next TDS switch with delay being any from 0 to Z TFs.

The case of $Z=K$ is called full forwarding (FF) since the incoming TF can be forwarded in any TF in one TC span. IF provides no freedom in selecting TF sequence at every switch along the route. Once a TF is selected at the first switch on the route all subsequent TFs are determined. Meanwhile, the case of FF is trivial for scheduling since it always brings valid schedules as long as resource is still available. Therefore, this work focuses on NIF, since it brings more scheduling flexibility and scalability, reducing blocking probability and increasing network utilization. However, the result of this work can also be easily applied for FF and IF cases.

B. TDS scheduling problem

Definitions:

Available TF - a TF at an output of a switch that have resource to carry packets of a requested flow.

Choice - a choice is an available output TF selected for a given flow for which a set-up request arrives at a switch. A choice is limited by the constraint: if at switch j, TF i ($0 \le i \le K$-1) is assigned, then at switch j+1, a TF in the range of [i, (i+Z)mod K] (in the same or next TC) can be a choice.

Schedule - a schedule is a sequence of choices over a predefined route of multiple switches.

Blocking of a schedule - a schedule is blocked at switch j when no choice is possible on that switch to advance the schedule to the next switch.

1. Network model: In TDS, routes are determined for any flow using existing routing protocols. TDS then focuses on the manner of (pipeline) forwarding the packets on that route. Hence, we will only study here the TDS problem: (1) on one predefined route (i.e, without route selection) with a predefined number of TDS switches as in Figure 2, (2) without propagation delay and alignment delay.

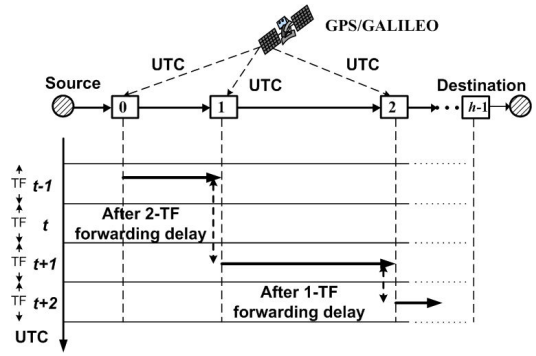

Figure 2 – Network model with $Z=2$-forwarding

The route is to carry traffic of the flow from Source to Destination (non-TDS points in our model), via h TDS switches.

2. Scheduling problem formulation: For *NIF*, sufficient bandwidth (available TFs) on every switch does not guarantee a non-blocking schedule to setup a flow, due to the mapping range restricted within Z TFs forwarding. TFs on a switch in general are assumed to be randomly available since the flows are stochastically created and released; thus all its available TFs may happen to be out of the Z-range for the considered request from the previous switch. Moreover, scheduling for NIF is a complicated task due to the larger size of the possible solution space, i.e., the large number of possible schedules as described below.

Observation 1: *For Z-forwarding* ($0 \le Z \le K$-1)*, (i.e., also IF and FF), with a single-channel, the total number of possible schedules for a h-switch route is:* $K \cdot (Z+1)^{h-1}$

Proof: There are K ***choices*** for the 0^{th} hop to forward a TF to the 1^{st} hop of the route. At the consecutive (h-1) hops, a TF can be buffered from 0 up to Z TFs before being switched.

Thus for all hop except the 0^{th} one, there are (Z+1) choices to schedule a TF. Since scheduling at all hops is independent (within the choice constraints), the total ***schedules*** are given by combination of all the possible single hop schedules:

$(K)_{0th} \cdot (Z+1)_{1th} \ldots (Z+1)_{h\text{-}1th} = K \cdot (Z+1)^{h-1}$

There is a big difference in complexity between NIF and IF which has only K possible schedules with K TFs per TC. For one flow, finding a proper schedule from a space of $K \cdot (Z+1)^{h-1}$ possibilities is a potentially complex problem due to the exponential boom of computations. A simple search for each request may take a long duration. Since we need to setup the flow (i.e., "virtual circuit") before the data transfer, the search time could restrict the reduction of the TF duration, and reduce the scalability of the TDS network (as the optical channel capacity increases, K and Z tend to increase as well). Moreover, with h in the exponent, the number of schedules grows exponentially with the size of the network.

Observation 2: *For Z-forwarding* ($0 \le Z \le K$-1)*, with C optical channels, the total number of possible schedules along a h-switch route is:* $K \cdot (Z+1)^{h-1} \cdot C^h$

Proof: the result is easily derived from the Observation 1

$(K \cdot C)_{0th} \cdot [(Z+1) \cdot C]_{1th} \ldots [(Z+1) \cdot C]_{h\text{-}1th} = K \cdot (Z+1)^{h-1} \cdot C^h$

Observation 1 and 2 show the exponential growth in the number of possible schedules. Thus, our objective is to design a scheduling algorithm that is efficient (low complexity) and robust, such that, it guarantees to find a schedule even if only one such schedule exists. Moreover, it may be important to minimize end-to-end delay. This, in turn, will lead to minimizing the required buffer size.

In general, a flow may be allocated a bandwidth of: (*i*) one TF, (*ii*) multiple TFs or (*iii*) a fraction of TF, thus the scheduling algorithm should be able to allocate such the schedules, accordingly. As a start point, in this work we focus on the one-TF scheduling case. We propose an algorithm based on the concepts from Viterbi algorithm [10], exploiting trellis diagrams. Thanks to their deterministic attributes, TDS resources on the route can be mapped to a suitable trellis structure to enable the search in dynamic stages. Also, from the analysis of different scenarios, the trellis will be adapted to each scenario with reasonable modifications. The scheduling algorithm is called **eSS** algorithm (**e**fficient-**S**urvivor-based-**S**earch) which finds the same best schedule returned by exhaustive search, while not suffering from the exponential growth in schedule options due to the deployment of a survivor-based mechanism. Within the scope of this paper, we address to the scenario of scheduling for a Single TF request in Single/Multiple-Wavelength Homogeneous TDS networks.

III. ESS - EFFICIENT-SURVIVOR-BASED-SEARCH

A. Underlying principles

A trellis diagram is used to describe all possible schedules set up in the TDS network for a given requested flow. The eSS algorithm then searches for the best schedule selection. Shortest delay will be used as the optimality criterion for schedule selection.

The trellis state diagram is constructed as follows (Figure 3):

- h ***stages*** present h switches.
- K ***states*** represent K TFs per TC of the output link of a switch on the route. Each T_i^j represents the binary status of ***state*** TF_i^j (TF i at switch j) to be considered for scheduling, where T_i^j is 0 for being Reserved or 1 for being Available. The states of the K TFs at each switch form a ***hop_availability_vector***: $\{T_i^j\} = S^j \times V^j$ (described in [2]), where:

  S^j: Switch availability vector: availability of each output of the switches along the route.

  V^j: Link availability vector: availability of each link along the route
- Each transition presents a feasible Z-forwarding between 2 states of 2 consecutive trellis stages, with its metric being the packet delay.
- Every trellis path is a sequence of trellis states on consecutive stages. A path represents a TF schedule for the flow. A path metric is the sum of all transition metrics on that path, which is the end-to-end delay of the flow.

The scheduling computation with the eSS algorithm is performed in a memoryless process: the computation at phase j is built up only on the computation result of the previous phase (j-1), without the necessity of referring to (j-2) and backward. This property is presented in our algorithm: Given the path metric (accumulated delay) up to an available state of the stage (j-1), the searching algorithm works on stage j based on joining the path metrics until *j-1* and the transition metric from *j-1* to j, then comparing metrics of all paths converging to a same state.

Only one best accumulated path, namely the "survivor", is kept per each state. This reflects the idea of searching for the best end-to-end forwarding delay schedule. The computation process is then progressed stage by stage until the destination stage is reached. At the last stage, at most K survivor paths, for K states respectively, would be available. The last comparison now among those K survivors yields the final selected schedule. Due to the memoryless searching property, the huge computational processing is avoided since the computation does not store and grow exponentially with the number of stages h.

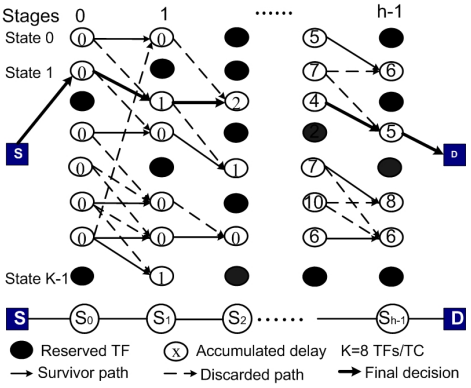

Figure 3 - An Example of Z= 2 (NIF), K= 8TFs per TC

The implementation can be done (*i*) in a distributed manner, with each switch computing its own result (for its trellis stage) and then passing the result to the next one; or (*ii*) in a centralized manner: one switch (or scheduling server) will take care of all the computations based on the information from all the other switches. Realizing the eSS algorithm needs some preliminary design settings: the availability of routing information and a signaling structure to transport the set-up messages.

B. Scheduling algorithm

The eSS scheduling algorithm can be formalized as follows:

Definitions:

- Path: a trellis curve from S to any state TF_i^j: $P_i = \{p_0,\ldots,p_j\}$ with $p_0,\ldots, p_j$ are state indices at stage 0,..,*j* and $p_j = i$
- Accumulated path metric (e.g., total accumulated delay) from S to TF_i^j : $\mu(P_i)$

A route of switches $\{S_0,\ldots,S_{h-1}\}$ is pre-determined by a routing algorithm.

Step 1: # With ***hop_availability_vector*** $\{T_i^0\}$ of switch 0:

1. **For** each TF *i*: *i*=0 to *K*-1
2. Initialize $P_i = \varnothing$
3. **If** $T_i^0 = 1$, put it to a new path $P_i = \{i\}$, $\mu(P_i)=0$

Step 2: # For each state *i* at stage (switch) *j*, select a survivor path that has the minimum accumulated path metric $\mu(P_i)$

4. **For** $\{T_i^j\}$ of each switch *j*: *j*=1 to *h*-1
5. **For** each TF *i*: *i*=0 to *K*-1
6. $P_i' = \varnothing$; $\mu(P_i') = \infty$
7. **If** $T_i^j = 1$
8. **For** $D_b = Z$ to 0
9. With TF *m*: $m=(i-D_b+K) \bmod K$
10. **If** $T_m^{j-1} = 1$ AND $P_m \neq \varnothing$
11. **If** $\mu(P_m) + D_b \leq \mu(P_i')$
12. $\mu(P_i') = \mu(P_m) + D_b$; $P_i' = P_m$

Store the path to prepare for the next iteration:

13. **For** each TF *i*: *i*=0 to *K*-1
14. **If** $P_i' \neq \varnothing$
15. $P_i = \{P_i', i\}$; $\mu(P_i) = \mu(P_i')$
16. **If** $P_i' = \varnothing$, then $P_i = P_i'$

Step 3:

17. After switch (*h*-1) finishes in step 2, find P_n ($0 \leq n \leq K-1$): $\mu(P_n) = \min_{0 \leq i \leq K-1} \mu(P_i)$. P_n is the best schedule.

Notice that the eSS algorithm minimizes the maximum buffering used at each switch by searching the survivor from "far" to "near" states (line 8-9), eSS replaces the current candidate with the path having smaller or equal metric. Hence, the final survivor (minimum-cost path) for each state contains the closest state from the previous stage, which results in minimum delay, hence buffering at that switch (stage).

In fact, the eSS algorithm is transparent to the bandwidth management of flows since the scheduling is per-request based. It means that the flow controller has freedom to decide if a schedule is per-flow, per-TF or even per TF-group basis by varying request rates to the source's scheduler. Hence eSS is flexible for any scheduling strategy: If scheduling is per-TF, the request is repeated for every TF of the flow. If per-flow scheduling is used, data of a flow follows the same scheduling after the first request during its whole span. The latter case has a lower computation compared to the former one; but bandwidth utilization might be lower if flows do not have continuous packets.

C. Proof of Integrity

Theorem

Let $\hat{P}$ be the best path from S to D found by eSS, and P^* the best path from S to D found by the exhaustive search for the same network, then $\hat{P} \equiv P^*$

Definition:

The set "*All*" of all possible paths from S to D can be divided into two sets:

- "*Discarded*": all the paths discarded during eSS
- "*Survived*" all survived paths up to D kept by eSS

Survived path at state TF_i^j : $sv(TF_i^j)$

Proof By definition of $\hat{P}$ and P^*

$\forall P: P \in$ "*Survived*" : $\mu(P) \geq \mu(\hat{P})$ (i)

$\forall P: P \in$ "*All*" : $\mu(P) \geq \mu(P^*)$ (ii)

If $P^* \in$ "Survived, (i) & (ii) can be merged and $\hat{P}$ means P^*.

Proof by contradiction: let's assume: $P^* \in$ "*Discarded*", i.e., is discarded by eSS at state $TF_{i^*}^{j^*}$. Hence P^* consists of 2 paths: from S to $TF_{i^*}^{j^*}$ (β_1) and from $TF_{i^*}^{j^*}$ to D (β_2). According to the eSS discarding rule:

$\exists\ sv(TF_{i^*}^{j^*})$: $\mu(sv(TF_{i^*}^{j^*})) < \mu(\beta_1)$ (iii)

Therefore,

$\exists\ P'$: $P' \in$ "*All*" and $P' = \{sv(TF_{i^*}^{j^*}), \beta_2\}$ (iv)

From (iii) and (iv) we can derive:

$\mu(P') = \mu(sv(TF_{j^*}^{i^*})) + \mu(\beta_2) < \mu(\beta_1) + \mu(\beta_2) = \mu(P^*)$

$\rightarrow \mu(P') < \mu(P^*)$ that contradicts to the definition of P^*.

Therefore, the Assumption is wrong, i.e., $P^* \in$ "*Survived*", and (i) & (ii) are then merged to say $\hat{P} \equiv P^*$

It is proved that the best-schedule resulting from the eSS algorithm (with discarding some paths on the way of forward dynamic programming searching) is the same as the one of the exhaustive search, yet avoiding the impractically exponential complexity as it has a linear complexity as shown in the following section.

D. Worst Case Complexity Analysis

1) *Exhaustive search:* Computation at each stage must take into account all the intermediate solutions produced by the computations at the previous stage. Thus, the computation cost is given:

$$X(h,K,Z) = K \cdot \left[\frac{(Z+1)^h - 1}{Z} - 1 \right] \quad h \geq 1 \qquad (1)$$

Obviously, the complexity of this $O(Z^h)$- algorithm grows exponentially with the size of the problem h, i.e., with the number of switches on a path.

2) eSS algorithm:

$$X(h,K,Z) = (h\text{-}1) \cdot K \cdot (Z+1) \quad h \geq 1 \qquad (2)$$

The maximum number of computation steps to obtain an optimal solution is linear in the size h of the problem. Thus, the complexity shows the eSS algorithm to be efficient to find out an optimal solution alike exhaustive approach, but with acceptable complexity.

Proof:

Stage 0: there is no transition metric computation: $C_0 = 0$

Stage 1: (Z+1) transitions are built for each of the K available states at stage 0; only the best path is kept for each state at stage 1: $C_1 = K \cdot (Z+1)$ and $S_1 = K$ paths.

Stage j: for each of the S_{j-1} paths retained at stage (j-1), (Z+1) possible transitions, hence new paths, can be created i.e., $C_j = S_{j-1} \cdot (Z+1) = K \cdot (Z+1)$, but only the best path is kept for each state at stage j, i.e., $S_j = K$. Considering a path of h switches, the scheduling complexity is:

$$X(h,K,Z) = \sum_{j=0}^{h-1} C_j = C_0 + \sum_{j=1}^{h-1} C_j = (h-1) \cdot K \cdot (Z+1)$$

E. eSS vs. Dijkstra algorithm

Starting from a list of vertices[1] (or states in the trellis diagram) for which the shortest path have been found, the well known Dijkstra algorithm [11][12] greedily considers all their neighbors (traditionally in the space domain) to add to the list the neighboring vertex reachable through the shortest path. This list updating is repeated until all vertices of the graph have been included, i.e., the shortest path to each of them has been found. Although the Dijkstra algorithm could in principle be deployed for the TDS scheduling, its execution cannot be easily performed cooperatively by the switches in a distributed manner. The Dijkstra best-fit approach could try to include in the above mentioned list a vertex corresponding to a TF in another switch. Hence, each step of the algorithm would require state information from various switches. The eSS algorithm, instead, by exploiting the topological structure of the trellis (in the time-space domain), carries out the search stage by stage (or, physically, switch by switch), keeping one path for each vertex (TF/state) in a stage. Consequently, the eSS algorithm can be naturally implemented in distributed manners over a route of TDS switches. Moreover, the eSS solution enables dynamic programming with less complexity. In the general implementation (linear search), the Dijkstra algorithm takes up to steps $O(V^2)$ for a graph $\{V,E\}$.Even for a sparse graph (e.g: not a full trellis, with small Z, making the number of edges small) in which the Dijkstra algorithm can utilize a priority queue with a binary heap, its complexity is $O((V+E)\log V)$ [12]. Both complexity figures are considerably greater than our solution's $O(E)$.

[1] The list originally includes only the vertex from which the shortest path tree is to be calculated.

(In our full trellis: the number of vertices is $V = h \cdot K$, and number of edges is $E \sim h \cdot K \cdot Z$).

IV. EXTENDING ESS TO WDM

TDS is working towards ultra-scalable switching and efficient bandwidth provisioning via being well coupled with WDM. The following extends the eSS scheduling algorithm to wavelength division multiplexing (WDM) in homogeneous TDS networks (i.e., where all optical channels have the same capacity). The scheduling algorithm in this case deals with the issue of wavelength and time frame assignment (WTA), which is related to wavelength and time-slot assignment with the major difference stemming from the nature of NIF (as specified using the parameter Z).

Assumptions and Definitions:

- C: WDM link capacity expressed as the number of optical wavelengths per optical fiber ($\lambda_1,\ldots,\lambda_C$).
- R: wavelength conversion range.

The scheduling feasibility in this case is related to the availability of capacity on a wavelength during a given time frame. Therefore, the objects dealt with by our scheduling algorithm here are a bi-dimensional resource, given by pairs of (TF_i,λ_m). In this context, the definitions of choice and schedule given in Section II.B and state given in Section III can be extended as follows:

State - $TF_i^j(\lambda_m)$ is TF i on λ_m at stage j

Choice **-** a choice is an available output TF on a wavelength selected for a packet flow for which a set-up request arrived at a switch. A choice is limited by the constraint: if at switch j , TF i is assigned ($0 \leq i \leq K$-1), then at switch j+1, a TF in the range of [i,(i+Z) mod K] (in the same or the next TC) on can be used.

Schedule - a schedule is a sequence of choices of a specific wavelength and a TF at each network switch, on a predefined route of multiple switches.

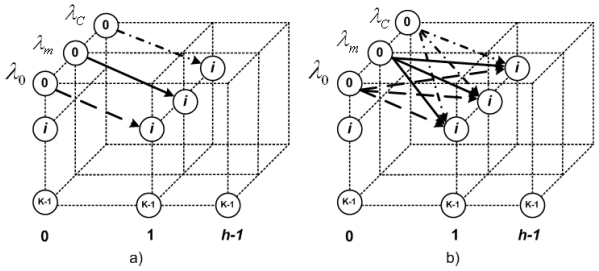

Figure 4 - Scheduling with a) no-wavelength-conversion, b) full wavelength conversion

Instead of the bi-dimensional trellis deployed in the previous basic case, a tri-dimensional one is required for the WDM case, as shown in Figure 4, features C planes, each representing one optical channel λ_m. Hence, the number of states N at each stage has grown C times with respect to the single-wavelength case, i.e., $N = C \cdot K$.

Applying the eSS algorithm to the WDM network, there is a usual but simple case in which **no wavelength conversion** is used just to extend the bandwidth by linear lambda allocation.

The scheduling problem can be seen as the combination of a wavelength assignment (WA) and time frame assignment (TA) sub-problem. Thus, we can run a known WA algorithm (first-fit, least-loaded etc.) [8] to select a wavelength first then TF searching on that lambda (disjoint approach). Another option (joint approach) is to select a wavelength based on specific information from the eSS algorithm, i.e., for the delay of the best schedule among the ones found previously on each lambda. With the joint solution, the eSS algorithm is to run C times in the worst case. With the disjoint solution in the worst case the algorithm must check all C planes before finding a schedulable one. Hence, in both cases the (worst case) complexity is C times the one of the eSS algorithm given by:

$$X(h,K,Z,C) = (h\text{-}1) \cdot K \cdot (Z+1) \cdot C \qquad (3)$$

In the case in which network switches have only **limited wavelength conversion** capabilities, by using tunable lasers [5], each wavelength can be converted to any wavelength in the range of R adjacent wavelengths, $R \leq C$, in a *contiguous wavelength selection fashion* ($R=C$ - full wavelength conversion).

1. Each state TF i is marked Available if the TF i is available on λ_m.
2. A valid transition exists between two available ***states*** $TF_i^j(\lambda_n)$ and $TF_k^{j+1}(\lambda_m)$ **iff**:
   (1) $D_b = (k - i + K) \bmod K \leq Z$
   (2) $W_b = |\lambda_m - \lambda_n| \leq R$
3. Perform searching as described in section III with forward-keeping at each state the path with minimum μ_b.

Metric μ_b can be constructed as a weighted sum of the multiple submetrics (i.e., delay, distance, and load). If the weight selected for one submetric, say delay, is much larger than the others, the schedule is selected according to the delay and the others are used only to select among equal-delay paths.

$$X(h,K,Z,R,C) = (h\text{-}1) \cdot K \cdot (Z+1) \cdot C \cdot R \qquad (4)$$

Proof: For each state, there are $(Z+1)\cdot R$ transitions to it from states of R planes, $(Z+1)$ states in each of R planes. So for $(C\cdot K)$ states, there are $(C\cdot K)\cdot[(Z+1)\cdot R]$ transitions at each stage. Thus, for h stages, the complexity is given:

$$X(h,K,Z,R,C) = (h\text{-}1) \cdot K \cdot (Z+1) \cdot C \cdot R$$

The algorithm is linear in size of h, K and Z. If $R \sim C$, we have quadratic complexity in the size of C since $O(C\cdot R) \sim O(C^2)$ is a polynomial complexity according to [9].

V. DISCUSSION

This work focused on the scenario of finding a schedule for a single TF requests in TDS networks. The proposed eSS (**e**fficient-**S**urvivor-based-**S**earch) scheduling algorithm is proven to be an efficient scheme that avoids the need to use either an exponential-complexity search or a heuristic algorithm. Furthermore, the eSS algorithm provides schedules with minimum delay.

Future works will include further algorithmic search design and analysis in various scenarios not considered in this work. In fact, different scheduling scenarios, as summarized in Table 1, can be formulated, depending on parameters such as:

Requested bandwidth per flow in TFs: a flow request may require bandwidth of a Single TF, Multiple TFs, or Fraction of one or more TFs to transport its traffic.

Number of optical channels: a link can contain a single-channel (SW) multiple channels (i.e. WDM with C lambdas multiplexed on one fiber on a link).

Network Type: when all links have the same capacity, having the same number K of TFs per TC, the TDS route is considered homogenous. Otherwise a TDS route is heterogeneous, with grooming and degrooming points.

Network Type \ Requested BW	Single TF	Multiple TFs	Fraction of TF
SW-Homogeneous	Case 1	Case 5	Case 9
WDM-Homogeneous	Case 2	Case 6	Case 10
SW-Heterogeneous	Case 3	Case 7	Case 11
WDM-Heterogeneous	Case 4	Case 8	Case 12

Table 1 – roadmap table

In Table 1, case 1 and case 2 are the two addressed scenarios of this paper. The primary challenge in our future work is to maintain the same level of search complexity for the remaining cases as the obtained result of the first two cases.

REFERENCES

[1] M. Baldi, Y. Ofek, "Fractional Lambda Switching," *Proc. of ICC 2002*, New York, vol.5.

[2] M. Baldi and Y. Ofek, "Fractional Lambda Switching - Principles of Operation and Performance Issues", *SIMULATION: Transactions of The Society for Modeling and Simulation International*, Vol. 80, No. 10, Oct. 2004.

[3] D. Grieco, A. Pattavina and Y. Ofek, "Flexible Bandwidth Provisioning in WDM Networks by Fractional Lambda Switching", *GlobeComm 2003*

[4] D. Grieco, A. Pattavina and Y. Ofek, "Fractional Lambda Switching for Flexible Bandwidth Provisioning in WDM Networks: Principles and Performance", Photonic Network Communications, Issue: Volume 9, Number 3, May 2005.

[5] V. T. Nguyen, R. Lo Cigno , Y. Ofek, "Design and Analysis of Tunable Laser-based Fractional Lambda Switching," IEEE INFOCOM 2006.

[6] M.Baldi, Yoram Ofek, "Grooming and Degrooming with Coordinated Universal Time (UTC)", SoftCOM 2003.

[7] N.F. Huang, G.H Liaw, C.P Wang, "A Novel All Optical Transport Network with Time Shared Wavelength Channels", *IEEE Journals on selected areas in communications*, Vol.18, No.10, October 2000.

[8] E. Karasan and E. Ayanoglu, "Effects of wavelength routing and selection algorithms on wavelength conversion gain in WDM networks", *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 186-196, Apr.1998.

[9] Michael Pinedo, "Scheduling – Theory, Algorithms and Systems", 2nd Edition, 2002, Prentice Hall Inc.

[10] Andrew J. Viterbi, "Error bound for convolutional codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Transactions on Information Theory* 13(2):260: April 1967.

[11] E. W. Dijkstra: "A note on two problems in connexion with graphs". In: *Numerische Mathematik*. 1 (1959).

[12] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, *Second Edition. MIT Press and McGraw-Hill*, 2001. ISBN 0-262-03293-7. Section 24.3: Dijkstra's algorithm, pp.595–601.