Dynamic survivable algorithm for meshed WDM optical networks

Lei Guo*, Hongfang Yu, Lemin Li

Key Lab of Broadband Optical Fiber Transmission and Communication Networks, University of Electronic Science and Technology of China, Chengdu 610054, PR China

Received 15 April 2005; received in revised form 28 July 2005; accepted 1 August 2005

Abstract

In this paper, we propose a novel heuristic survivable algorithm called dynamic path-shared protection (DPSP) to completely protect the double-link failures in meshed Wavelength-division-multiplexing (WDM) optical networks. In order to improve the algorithm performance, we focus on considering two key issues that are load balancing and resource-sharing degree when computing the working and backup paths. We also investigate the trap situations and present a solution method, because the trap situations may lead to high blocking probability (BP). Simulations results show that, DPSP can provide complete protection for the double-link failures; with respect to the previous work, DPSP not only can effectively avoid the trap situations but also is able to obtain higher resource utilization ratio and lower BP.

© 2005 Elsevier Ltd. All rights reserved.

Keywords: Wavelength-division-multiplexing (WDM); Optical networks; Double-link failures; Path-shared protection; Trap situations; Load balancing; Resource-sharing degree

*Corresponding author. Also at: College of Information Science and Engineering, Northeastern University, Shenyang 110004, China.

E-mail addresses: lguo@uestc.edu.cn, guolei@ise.neu.edu.cn (L. Guo).

1084-8045/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2005.08.006
1. Introduction

In wavelength-division-multiplexing (WDM) optical networks, a wavelength channel has the transmission rate over gigabits per second (Mukherjee, 1997). If fiber links fail, a lot of connection streams would be dropped. Therefore, the protection design for WDM optical networks is very important. Previous work has studied the single-link failure problem (Ramamurthy et al., 2003; Ou et al., 2004; Ho et al., 2004; Ho and Mouftah, 2004; Wen et al., 2003; He et al., 2004) that is dominant in WDM optical networks. Recently, with the network size keeping enlarging, the multiple-link failures probability becomes much higher. Therefore, the protection design for the double-link failures must be considered (Choi et al., 2002; Kim and Lumetta, 2003; Schupke and Prinz, 2003; He and Somani, 2003; Zhang et al., 2004; Jozsa et al., 2003; Guo et al., 2005).

Some papers have investigated the double-link failures problem and presented the protection algorithm (He and Somani, 2003; Zhang et al., 2004; Jozsa et al., 2003; Guo et al., 2005). The algorithm in (He and Somani, 2003) is for static network traffics, so that it is not suitable for dynamic network traffics because it uses the integer-linear-programming (ILP) method that is very complicated. Another algorithm in Zhang et al. (2004) is for dynamic network traffics, but it is unable to provide complete protection for the double-link failures. The two heuristics presented in Jozsa et al. (2003), Guo et al. (2005) are both for dynamic network traffics, and they are also able to provide complete protection for the double-link failures, so that they are suitable for on-line computation. For each connection request, the two heuristics first compute a working path, second compute the first link-disjoint backup path, and third compute the second link-disjoint backup path. Therefore, the two heuristics are called three-step-approach (TSA). A drawback of TSA is that, although it chooses the shortest routes as the working and backup paths, the algorithm performance (e.g., resource utilization ratio (RUR), blocking probability (BP), etc.) may not be good (Guo et al., 2004a). Another limitation of TSA is that, it cannot find a solution in trap situations, even though a solution exists. Then more connection requests may be blocked in trap situations, and the BP may be high (Papadimitriou et al., 2001; Oki et al., 2002).

In order to overcome the two drawbacks of TSA, in this paper we propose a novel heuristic survivable algorithm called dynamic path-shared protection (DPSP) to completely protect the double-link failures in meshed WDM optical networks. When computing the paths, we present three link-cost functions to adjust the load balancing and the resource-sharing degree according to the current network state, and we also develop a solution method to avoid the trap situations. Therefore, according to simulation results in Section 4, we observe that, DPSP can provide complete protection for the double-link failures; with respect to the previous work, DPSP not only can effectively avoid the trap situations but also is able to improve the RUR and reduce the BP.

If we consider the shared-risk link group (SRLG) constraints (Xu et al., 2003), those algorithms presented in Ou et al. (2004), Ho et al. (2004), Ho and Mouftah (2004), Wen et al. (2003), He et al. (2004) for protecting the single-link failure can be
easily extended to protect the single-SRLG failure (Papadimitriou et al., 2001; Oki et al., 2002; Yu et al., 2003), and those algorithms presented in He and Somani, (2003), Zhang et al. (2004), Jozsa et al. (2003), Guo et al. (2005) and our algorithm for protecting the double-link failures also can be easily extended to protect the double-SRLG failures (Guo et al., 2004b).

The rest of the paper is organized as follows. Section 2 states the network model, backup resources assignment, and the link-cost functions. Section 3 describes the trap situations and our solution method. Section 4 presents the performance parameters and simulation results. Section 5 concludes this paper.

2. Problem definition

2.1. Network model

The network topology is \( G(N, L, W) \) for a given meshed WDM optical network, where \( N \) is the set of nodes, \( L \) is the set of bi-directional links, and \( W \) is the set of available wavelengths per fiber link. \( |N|, |L| \) and \( |W| \) denote the node number, the link number and the wavelength number per fiber link, respectively. Connection requests arrive at the network dynamically, and there is only a connection request arrives at a time. We assume each connection requires a wavelength channel bandwidth, and assume each network node has the full wavelength conversion \((OEO)\). A shortest-path algorithm, Dijkstra’s algorithm, is applied to compute the routes. The following notations are introduced:

\( l \): bi-directional fiber link in \( G \).
\( c_f \): basic cost for link \( l \). It is determined by many factors, such as the physical length of fiber link, installation cost of fiber link, and so on.
\( c_f' \): dynamic cost for link \( l \). It is determined by \( c_f \) and the current network state.
\( WP_n \): working path for connection request \( n \).
\( BP_1^n \) and \( BP_2^n \): first and second backup paths for \( WP_n \), respectively.
\( W_l, F_l \) and \( R_l \) are illustrated in Fig. 1, where \( W_l \) is the total number of working wavelength links on link \( l \), \( F_l \) is the total number of free wavelength links on link \( l \), \( R_l \) is the total number of reserved backup wavelength links on link \( l \).

![Fig. 1. An illustration of capacities distribution for link \( l \).](image-url)
$v^e_l$: set of connections whose working paths traverse link $e$ and the corresponding backup paths traverse link $l$.
$|\Omega|$: number of elements in set $\Omega$.
$FR_l$, $SR_l$, $RFR_l$, and $RSR_l$: variables used in our algorithm.

2.2. Reserved backup resources assignment

We assume the connection request $n$ arrives at a given time. For arbitrary link $l$, we can obtain $FR_l = \max(|v^e_l|)(\forall e \in L, e \neq l)(RFR_l = v^e_l)$ and $SR_l = \max(|v^e_l - v^e_l \cap RFR_l|)(\forall t \in L, t \neq l, t \neq e)(RSR_l = v^e_l - v^e_l \cap RFR_l)$. Therefore, the reserved backup resources $R_l = FR_l + SR_l$. We give an illustration in Fig. 2.

In Fig. 2, we note that there are six working paths whose second backup paths all traverse link $l$. First, we obtain $v^x_l = \{0, 1, 2\}$, $v^y_l = \{3, 4\}$, and $v^z_l = \{5\}$. Second, we get $FR_l = |v^l_x| = 3$ and $RFR_l = v^l_x = \{0, 1, 2\}$. Finally, we get $SR_l = |v^l_y - v^l_x \cap RFR_l| = 2$ and $RSR_l = v^l_y - v^l_x \cap RFR_l = \{3, 4\}$. Then, the reserved backup resource $R_l$ is equal to five. In the worst case, if links $x$ and $y$ fail simultaneously, the working paths $WP_0-WP_4$ and their first backup paths all fail, because the first backup paths of working path $WP_0-WP_2$ traverse link $y$ and the first backup paths of working paths $WP_3$ and $WP_4$ traverse link $x$. The traffics of these working paths all can be switched to their second backup paths that all traverse link $l$, and then five reserved backup wavelength links on link $l$ are enough to transmit the switched traffics. Thus, these connection traffics can all be protected effectively.

![Fig. 2. An illustration for reserved backup resources assignment.](image-url)
2.3. Link-cost assignment

2.3.1. Load balancing

We assume the connection request \( n \) arrives at a given time. Before computing the working path, we can adjust the link-cost according to Eq. (1):

\[
c_l^\prime = \begin{cases} 
    +\infty & \text{if } F_l = 0, \\
    -\alpha F_l + c_l & \text{otherwise.}
\end{cases}
\]  

(1)

In Eq. (1), \( \alpha \) is a positive constant considering load balancing (note that \( -\alpha F_l + c_l \) must be greater than zero). It is obvious that, when \( \alpha \) increases, the links that have more free resources (i.e., bigger \( F_l \)) will have less link-cost. Then, the working paths will be favorable for traversing these links, and the consumed resources can be more uniformly distributed to all links. Thus, the load may be more balance and the request \( BP \) may be decreased. We give an illustration in Fig. 3.

In Fig. 3(b), there already exist two reserved backup wavelength links on link \( d-b \). For a new connection request with source node \( d \) and destination node \( b \), (1) if we use the shortest-path algorithm, we can find the working path \( d-b \) and two link-disjoint backup paths \( d-c-b \) and \( d-a-b \) in Fig. 3(c), and then five wavelength links (a working wavelength link and four backup wavelength links) need to be assigned; (2) if we use our least-cost path algorithm considering the load balancing according to Eq. (1), we can find the working path \( d-a-b \) and two link-disjoint backup paths \( d-b \) and \( d-c-b \) in Fig. 3(d), and then only four wavelength links (two working wavelength links and two backup wavelength links) need to be assigned because a reserved wavelength link existing on link \( d-b \) can be shared (see Fig. 3(d)). It is obvious that the algorithm with load balancing can save more resources, and then more free resources can be used by the following connection requests. Therefore, fewer connection requests may be blocked and the \( BP \) can be decreased.

2.4. Backup resource-sharing degree

If \( WP_n \) has been found, we can adjust the link-cost according to Eq. (2) before computing the first link-disjoint backup path. In Eq. (2), \( \varepsilon \) is a positive constant.
considering the resource-sharing degree (note that $-e|W| + c_l$ must be greater than zero) and $U = \{l; l \leq WP_n\}$. After finding the first backup path $BP_n^1$, we can adjust the link-cost according to Eq. (3) before computing the second link-disjoint backup path. In Eq. (3), $Q = U + \{l; l \in BP_n^1\}$:

$$c'_l = \begin{cases} +\infty & \text{if } (l \cap U \neq \emptyset) \cup (F_l + R_l < FR_l + SR_l), \\ -e|W| + c_l & \text{if } R_l \geq FR_l + SR_l, \\ c_l & \text{otherwise}. \end{cases}$$

(2)

$$c'_l = \begin{cases} +\infty & \text{if } (l \cap Q \neq \emptyset) \cup (F_l + R_l < FR_l + SR_l), \\ -e|W| + c_l & \text{if } R_l \geq FR_l + SR_l, \\ c_l & \text{otherwise}. \end{cases}$$

(3)

We can see from Eqs. (2) and (3) that, these links that already have enough reserved backup resources (i.e., $R_l \geq FR_l + SR_l$) have less link cost. If the backup paths traverse these links, we need not reserve new backup resources. Thus, the $RUR$ will be improved. We give an illustration in Fig. 4.

In Fig. 4(b), there already exist two reserved backup wavelength links on links $f-a$ and $a-b$. For a working path $e-c$, (1) if we use the shortest-path algorithm, we can find the two backup paths $e-f-b-c$ and $e-d-c$ in Fig. 4(c), and then five reserved backup wavelength links need to be assigned; (2) if we use the least-cost routing algorithm considering the backup resource-sharing degree according to Eqs. (2) and (3), we can find the two backup paths $e-f-a-b-c$ and $e-d-c$ in Fig. 4(d), and then only four reserved backup wavelength links need to be assigned because the two reserved wavelength links existed on links $f-a$ and $a-b$ can be shared (see Fig. 4(d)). It is obvious that the algorithm with backup resource-sharing degree can save more resources, and more free resources can be used by the following connection requests. Thus, fewer connection requests may be blocked and the $BP$ can be decreased.

Fig. 4. An illustration for sharing reserved resources: (a) a network topology; (b) there already exist two reserved wavelength links on links $f-a$ and $a-b$; (b) wavelength links assignment with shortest-path for a connection request (source node $e$, destination node $c$); (d) wavelength links assignment with sharing reserved resources consideration.
3. Trap situations and proposed algorithm

Previous algorithm TSA is a three-step algorithm, which first computes a working path, second computes the first link-disjoint backup path and third computes the second link-disjoint backup path. A limitation of TSA is that it cannot find a solution in trap situations, even though the solution exists. Then more connection requests may be blocked in trap situations, and the BP may become high. In the following, we will present two trap cases and suggest our solution method.

3.1. Link-disjoint caused trap

A trap case is called Link-disjoint caused trap that is shown in Fig. 5. It is obvious that TSA cannot find three link-disjoint paths for a connection request with source node 1 and destination node 5, even though they all exist. TSA only can find the working path 1–2–3–4–6–5, and then the request will be blocked in this situation. Now, we present our backtracking method based on the network information to overcome the trap situations.

We refer to a link on the working path as a backhaul link. For example, in Fig. 5(a), link 1–2 is a backhaul link. All backhaul links on the working path compose the set $L_w$. If we increase the cost of the backhaul link 1-2 to sufficient large (i.e., 1000) in Fig. 5(b), a new working path 1–3–4–6–5 and the first link-disjoint backup path 1–2–5 can be both found. Based on the new working path, we recompose the set $L_w$. In Fig. 5(b), however, we cannot find the second link-disjoint backup path. We refer to a link on the first backup path as a backhaul link. All backhaul links on the first backup path compose the set $L_b$. We increase the cost of any backhaul link in $L_b$ and re-compute the first backup path. However, the second link-disjoint backup path cannot be found yet, even though we increase all links costs in $L_b$. Therefore,

Fig. 5. Link-disjoint caused trap; solid lines represent the bi-directional fiber links; dashed lines denote the paths; and the numbers besides the links represent the link-cost: (a) If the working path is 1–2–3–4–6–5, no backup paths can be found. (b) If the working path is 1–3–4–6–5, only a backup path 1–2–5 can be found. (c) If the working path is 1–6–5, two backup paths 1–2–5 and 1–3–4–5 can be both found.
we need backtrack to $L_w$ and increase the cost of any backhaul link in $L_w$. If we increase the cost of the backhaul link 1–3 to sufficient large (i.e., 1000), a new working path 1–6–5 and two link-disjoint backup paths 1–2–5 and 1–3–4–5 can all be found in Fig. 5(c). Thus, the trap situations can be effectively avoided.

3.2. Resource-sharing caused trap

Another case of trap situations is termed Resource-sharing caused trap, which is illustrated in Fig. 6. A connection, whose working path 5–4–3 and two link-disjoint backup paths 5–6–3 and 5–1–2–3, has existed in the network (other existing connections are not shown). Now, we assume that all links have enough free resources except that link 5–1 such that the free resource $F_{5-1}$ is equal to zero. For a new connection request with source node 5 and destinations node 6, TSA only can find the working path 5–4–6 and a link-disjoint backup path 5–6. In this situation, the potential backup path is 5–1–2–6. Because the two working paths, which are 5–4–3 and 5–4–6, are not link-disjoint on link 5–4, then the reserved backup resources on link 5–1 cannot be shared. Owing to $F_{5-1}$ is equal to zero, so that no free resources can be assigned and the potential backup path 5–1–2–6 cannot be established. Therefore, we need consider avoiding the trap problem.

We define a link $e$ as a conflicting link if there exist another links $j$ and $d$ such that $|v_e^j| + |v_e^j \cap v_e^d| = R_j$ and $F_j = 0$. All conflicting links on working path compose the set $L_c$. For example, if $|v_{5-1}^{5-4}| + |v_{5-1}^{5-4} \cap v_{5-1}^d| = R_{5-1}$, $\forall d \neq 5-4$ and $F_{5-1} = 0$, we obtain link 5–4 is a conflicting link. If we cannot find the backup path and the conflicting links exists, we increase the cost of the conflicting link to some sufficient large value and restart the computing process. For example, if we increase the cost of the conflicting link 5–4 o 1000 in Fig. 5(c), we can find a working path 5–6 nd two link-disjoint backup paths 5–4–6 and 5–1–2–6. Therefore, the trap situation can be effectively avoided.
3.3. Proposed algorithm

Because there possibly exist chained \textit{trap} situations, in which some \textit{traps} do not occur until some others are processed, so that we can recursively apply this procedure. We give a parameter \(K\) to limit the number of recursions. If we find a feasible solution within \(K\) limitation, the procedure will be stopped. The process of \textit{DPSP} is presented as follows.

\textbf{Dynamic path-shared protection (DPSP)}

\textbf{Input:} \(G = (N, L, W);\) a connection request \(n;\) source node, destination node \(\in N;\) \(c \leftarrow 0;\) \(K\)

\textbf{Output:} Three link-disjoint paths \((WP_n, BP_n^1, BP_n^2)\), or NULL if no paths are found.

\textbf{Step 1:} Let \(WP_n \leftarrow \text{NULL}, BP_n^1 \leftarrow \text{NULL}, L_w \leftarrow \text{NULL}, L_b \leftarrow \text{NULL},\) and \(L_c \leftarrow \text{NULL}.\) Go to Step 2.

\textbf{Step 2:} Adjust the link-cost according to Eq. (1), increase the cost of any link in \(L_w\) or \(L_c\) to some large value, and compute the least-cost working path.

If succeed to find the working path and \(WP_n \neq WP_n',\) go to Step 3.

Else, abandon the request and return NULL.

\textbf{Step 3:} Adjust the link-cost according to Eq. (2), increase the cost of any link in \(L_b\) to some large value, and compute the first least-cost backup path.

If succeed to find the first backup path and \(BP_n^1 \neq BP_n^1,\) go to Step 4.

Else, compose the set \(L_w\) of the \textit{backhaul links} and the set \(L_c\) of the \textit{conflicting links}.

Let \(c \leftarrow c + 1\) and \(WP_n' \leftarrow WP_n.\)

If \(c < K,\) let \(BP_n^1 \leftarrow \text{NULL}, L_b \leftarrow \text{NULL}.\) Go back to Step 2.

Else, abandon the request and return NULL.

\textbf{Step 4:} Adjust the link-cost according to Eq. (3), and compute the second least-cost backup path.

If succeed to find the second backup path, go to Step 5.

Else, compose the set \(L_b\) of the \textit{backhaul links} and the set \(L_c\) of the \textit{conflicting links}.

Let \(c \leftarrow c + 1, WP_n' \leftarrow WP_n\) and \(BP_n^1 \leftarrow BP_n^1.\)

If \(c < K,\) go back to Step 3.

Else, abandon the request and return NULL.

\textbf{Step 5:} Record the routing and resources assignment information. Return \((WP_n, BP_n^1, BP_n^2).\)

Above procedure shows that, if \(K \leq 1,\) the algorithm may not avoid the \textit{trap} situations by recursively applying this procedure, and then it is equivalent to the three-step approach. So, in order to avoid the \textit{trap} situations, \(K > 1\) should be satisfied (in simulations we set \(K = 10).\)

The time complexity of \textit{DPSP} mostly depends on running the times of Dijkstra’s algorithm whose time complexity is \(O((|N| + |L|)\log(|N|))\) for graphs with weighted vertices (Barbehenn, 1998). Analyzing the process of the algorithm, in the worst case, \textit{DPSP} will run one time of Dijkstra’s algorithm to compute a working path and run one time of Dijkstra’s algorithm to compute the first backup path for a connection request; if the second backup path cannot be found, \textit{DPSP} will mostly run \(K\) times of Dijkstra’s algorithm to
re-compute the routes. So the time complexity of DPSP is approximately \( O((2+K)(|N|+|L|)\log(|N|)) \).

4. Simulations and analysis

4.1. Performance parameters

The load balancing degree (LBD) is written as

\[
LBD = \frac{|L| \max \{W_i\}}{\left( \sum_{i \in L} W_i \right)}. \tag{4}
\]

It is obvious that the LBD is close to 1 means that the traffic loads are more uniformly distributed to the network, and the algorithm is more favorable for the load balancing.

The RUR is written as

\[
RUR = \frac{\left( \sum_{i \in L} R_i \right)}{\left( \sum_{i \in L} W_i \right)}. \tag{5}
\]

It is obvious that a smaller value of the RUR means that we need to assign fewer resources, and also means a smaller backup resources reserve on all the backup paths and a higher degree of reserved resources sharing, that is, a higher RUR. Higher RUR will lead to lower traffic blocking because more free resources can be used by the following connection requests.

The BP is the ratio of \( R \) to \( V \), where \( R \) is the total number of connections that are being abandoned by the network and \( V \) is the total number of all connection requests that have arrived at the network. In the case of dynamic traffic, the BR can approximately reflect the effectiveness of resource utilization, and a smaller BR means a higher RUR.

The Dropping ratio (DR) is the ratio of \( D \) and \( A \), where \( D \) is the total number of connections that are dropped when link failures occur, and \( A \) is the total number of all connections holding on the network. The DR is equal to zero means that the algorithm can provide the complete protection.

4.2. Testing model

We simulate a dynamic network environment with the assumptions that connection requests arrive according to an independent Poisson process with arrival rate \( \beta \) and that the connections’ holding times are negatively exponentially distributed, \( 1/\mu \); that is, the network load is \( \beta/\mu \) erlang. We assume \( \mu \) is equal to 1 and each connection requires a wavelength channel bandwidth. If the connection fails to be established, the network abandons it immediately; i.e., there are no waiting queues. The test network is shown in Fig. 7. We assume each network node has the full wavelength conversion (OEO), and assume each node pair is
interconnected by a bi-directional fiber link whose basic link cost is assumed to be 100. The number of wavelengths per fiber is assumed to be 20. We compare the performances of the DPSP with the previous TSA (Jozsa et al., 2003; Guo et al., 2005). In simulations, we set $K = 10$ for DPSP in order to avoid the trap situations. All simulation results are averaged by simulation of $10^6$ connection requests.

4.3. Results analysis

Fig. 8(a) shows that the LBD is big for $z = 0$, and reduces and gradually becomes invariable as $z$ increases. The reason for this is that there is no consideration about the load balancing according to Eq. (1) as $z = 0$. When $z$ increases, the links that have more free resources will have less link-cost. Then, the working paths will be favorable for traversing these links, and the consumed resources may be more uniformly distributed to the network. Thus, the load can be more balancing.
Fig. 8(b) shows that the RUR reduces and gradually becomes invariable as $\varepsilon$ increases. Because as $\varepsilon$ increases, according to Eqs. (2) and (3), the links that already have enough reserved resources will have less link-cost. Then, the backup paths will be favorable for traversing these links, that is, they are favorable for selecting the least additional reserved backup resources as the paths. Thus, the RUR can be improved.

Based on above analysis, in the following discussion, we choose $(a, e, K) = (0, 0, 10)$ and $(a, e, K) = (4, 4, 10)$ for DPSP. It is obvious that, when $(a, e, K) = (0, 0, 10)$, DPSP cannot adjust the load balancing and the backup resource-sharing degree but can avoid the trap situations; when $(a, e, K) = (4, 4, 10)$, DPSP not only can adjust the load balancing and the backup resource-sharing degree but also is able to avoid the trap situations. In fact, accordance with above analysis (see Sections 2.3 and 3.3), TSA is a special case of DPSP when $(a, e, K) = (0, 0, 1)$.

Fig. 8(c) shows that, when two random links fail, the DRs of DPSP and TSA both are equal to zero, and this means that they can both completely protect the double-link failures.

Fig. 8(d) shows that, when $(a, e) = (4, 4)$, DPSP obtains better RUR than TSA with different load. The reason for this is that, the DPSP that computes the least-cost paths considers the load balancing and the backup resource-sharing degree. Therefore, when $(a, e) = (4, 4)$ DPSP yields higher RUR than TSA (the reasons for this can be found in Section 2.3), and this will lead more free resources can be used by the following connection requests. Therefore, when $(a, e) = (4, 4)$ DPSP has a lower BP than TSA, which has been shown in Fig. 8(d).

Fig. 8(d) also shows that, when $(a, e) = (0, 0)$ the performance of RUR for DPSP is near TSA, because when $(a, e) = (0, 0)$ DPSP cannot adjust the load balancing and the resource-sharing degree according to the current network state, and the paths for DPSP and TSA are all the shortest routes. However, in Fig. 8(e), we observe that, when $(a, e) = (0, 0)$ DPSP obtains lower BR than TSA. The reason for this is that DPSP with $K = 10$ can avoid the trap situations and is able to reduce the traffic blocking but TSA cannot. This means that, compare to TSA, DPSP can effectively avoid the trap situations.

Fig. 8(e) also shows that, when $(a, e) = (4, 4)$ DPSP has the lowest BP. A reason for this is that DPSP with $(a, e) = (4, 4)$ has higher RUR than DPSP with $(a, e) = (0, 0)$, and another reason is that DPSP can effectively avoid the trap situations but TSA cannot.

We can thus conclude that, DPSP can provide complete protection for the double-link failures; with respect to the previous work, DPSP not only can effectively avoid the trap situations but also is able to yield higher RUR and lower BP.

5. Conclusions

In this paper, we studied the double-link failures problem and proposed a novel dynamic protection algorithm called DPSP for meshed WDM optical networks. In order to improve the performance, we focused on considering the load balancing and
the backup resource-sharing degree according to the current network state. We also investigated the trap situations and presented our solution method, because the trap situations may lead to high BP. Simulations results showed that DPSP outperforms the previous TSA.

Acknowledgments

This work was supported by National Science Foundation of China (NSFC) under Grants 60302010. The authors would like to thank anonymous reviewers for valuable comments.

References


