Evolutionary Algorithms Refining a Heuristic: A Hybrid Method for Shared-Path Protections in WDM Networks Under SRLG Constraints

Qingfu Zhang, *Member, IEEE*, Jianyong Sun, Gaoxi Xiao, *Member, IEEE*, and Edward Tsang, *Member, IEEE*

***Abstract*—An evolutionary algorithm (EA) can be used to tune the control parameters of a construction heuristic to an optimization problem and generate a nearly optimal solution. This approach is in the spirit of indirect encoding EAs. Its performance relies on both the heuristic and the EA. This paper proposes a three-phase parameterized construction heuristic for the shared-path protection problem in wavelength division multiplexing networks with shared-risk link group constraints and applies an EA for optimizing the control parameters of the proposed heuristics. The experimental results show that the proposed approach is effective on all the tested network instances. It was also demonstrated that an EA with guided mutation performs better than a conventional genetic algorithm for tuning the control parameters, which indicates that a combination of global statistical information extracted from the previous search and location information of the best solutions found so far could improve the performance of an algorithm.**

***Index Terms*—Estimation of distribution algorithms (EDAs), evolutionary algorithm (EA), guided mutation, hyperheuristics, memetic algorithm (MA), network protection, shared-risk link group (SRLG).**

I. Introduction

THE COMBINATION of evolutionary algorithms (EAs) and problem-specific heuristics has been proven very successful in dealing with hard search and optimization problems. The majority of current combination schemes adopt one or more of the following three related approaches.

1) Memetic algorithm (MA) [1]–[8]: MAs are inspired by cultural evolution. They employ one or several problem-specific heuristics to improve and/or repair some or all of the solutions generated by EA offspring generators (such as crossover and mutation). In a canonical MA, a single heuristic local search procedure is applied to every newly generated solution, while adaptive MAs use a number of heuristics, and the decision on which heuristic to improve a new solution is made dynamically. Therefore, adaptive MAs allow both cooperation and competition among different heuristics. A very recent classification of adaptive MAs based on adaptation level can be found in [9].
2) Hyperheuristic [10]–[12]: This is a general methodology in search and optimization. Typically, a hyperheuristic does not directly work on the solution space of the optimization problem. Instead, it manages a set of knowledge-poor and low-level heuristics that can modify or repair an existing solution to the problem. At any given decision point, high-level heuristics choose a low-level heuristic and then apply it to the solutions obtained from the previous stage of search. A hyperheuristic can use an EA as its high-level heuristic for searching good low-level heuristics. Portfolio algorithms, based on ideas from economics, also aim at combining different heuristics for solving hard optimization problems [13].
3) Indirect encoding EAs using construction heuristics [5], [14]–[17]: Suppose that we have a problem-specific construction heuristic for an optimization problem. The heuristic has a set of control parameters with relatively simple data structures. For any given parameter setting, the heuristic can construct a solution to the problem, the quality of which entirely depends on the parameter setting. An EA can be used to tune these control parameters to find a nearly optimal parameter setting and thus generate a good solution to the original problem. This approach can be regarded as an instance of indirect encoding EAs since the actual search space of the EA is the parameter space of the heuristic and each parameter setting can be decoded (i.e., transformed) to a solution to the original problem via the heuristic. If solutions to the problem have a complex data structure, and a parameterized construction heuristic is relatively easy to design, this approach could be a reasonable choice.

Conventional EAs [18]–[20] mainly employ crossover and mutation for generating new solutions. Generally, only a few (often two in crossover and one in mutation) parent solutions are directly involved in these operators. There is no mechanism in conventional EAs for extracting global statistical information from the previous search and using it for guiding the further search. Estimation of distribution algorithms (EDAs) [21]–[24] work in a quite different way: they maintain a probability model for characterizing the distribution of promising solutions at each generation. The model is updated based on the global

Manuscript received April 26, 2005; revised October 10, 2005 and February 6, 2006. This paper was recommended by Guest Editor Y. S. Ong.

Q. Zhang and E. Tsang are with the Department of Computer Science, University of Essex, CO4 3SQ Colchester, U.K.

J. Sun is with the School of Computer Science, University of Birmingham, B15 2TT Birmingham, U.K.

G. Xiao is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2006.883269

statistical information extracted from the current population. New solutions are generated by sampling from this model. However, information about the location of the best solutions found so far has not been directly utilized. Recently, we have proposed a new operator called guided mutation [25], [26] for generating new solutions in EAs. Guided mutation combines the global statistical information and location information of the solutions found so far so that the shortcomings of EAs and EDAs are efficiently overcome.

In this paper, an indirect encoding EA using a construction heuristic is proposed for the shared-path protection (SPP) problem in wavelength division multiplexing (WDM) optical networks under shared-risk link group (SRLG) constraints [27]. This problem requires finding a working lightpath and a backup lightpath for each of the set of connection requests with the objective of minimizing the wavelength capacities consumed in these lightpaths. Each feasible solution to this problem, therefore, consists of a set of working lightpaths and a set of backup lightpaths subject to a large number of constraints. If we encode a solution as a binary string, the resultant problem size, for a practical-sized network, would become prohibitively large to deal with [27]. It is also not an easy task to design EA operators such as crossover and mutation for operating directly on these solutions. Therefore, the commonly used framework of MAs is not very suitable for this problem. In our proposed method for this problem, a three-phase parameterized construction heuristic is used to construct a solution. The first and second phases are for constructing working lightpaths and backup lightpaths, respectively. The third phase assigns a wavelength to each lightpath. The quality of the solution generated in the construction heuristic is determined by its control parameters: two permutation vectors π and σ, and a real parameter c. An EA with guided mutation (EA/G) and a genetic algorithm (GA) are employed for optimizing π and σ, while c is tuned by testing its several representative values. The experimental results show that EA/G outperforms the conventional GA in tuning these control parameters, which indicates that the combination of location information and global statistical information can improve the performance of an EA. The results also show that our proposed approach performs significantly better than the heuristic of Zang–Ou–Mukherjee [27].

This paper is organized as follows. Section II describes in detail the SPP problem in WDM optical networks under SRLG constraints. Section III presents the construction heuristic for the SPP problem. Sections IV and V introduce GA and EA/G, which are used for tuning the control parameters. The framework of the proposed approach for the SPP problem is given in Section VI. Section VII presents the experimental results. Finally, Section VIII concludes this paper.

II. Problem Definition

In this paper, we consider the network protection problem in WDM networks. Due to their capability of efficiently utilizing the huge bandwidth of optical fibers, WDM networks are now the mainstream solution for supporting high-speed long-distance communications. In such high-speed networks, however, a single link failure can lead to serious service interruptions. Therefore, network protection is of critical importance. Among the most popular protection methods is the so-called path protection [28], where we set up two link-disjoint lightpaths (named as the working lightpath and the backup lightpath, respectively) between each source–destination pair. Here, a lightpath is a directed path from source to destination along which all the links use the same wavelength. To save on network capacity, different backup lightpaths can share the same wavelength channel as long as their respective working lightpaths are not going through any common link. Extensive researches have been carried out on investigating this problem (e.g., [29]–[31]).

One of the most important recent developments in WDM network protection is the introduction of the SRLG concept [32], [33]. SRLG is defined as a set of network components with a significant probability of failing simultaneously, e.g., fibers going through the same duct. Although considering SRLGs help to strengthen the network survivability, it makes the network protection problem much more difficult mainly because of the two sets of additional constraints it imposes, namely 1) a working lightpath and its backup lightpath cannot go through any common SRLG (whereas in the classic network protection problem, we only have to ensure that they do not go through any common link); and 2) if two working lightpaths go through a common SRLG, their backup lightpaths cannot share any network resource (while in the classic protection problem, similar constraints only apply to working paths going through the same link).

The SPP problem can be modeled as an optimization problem in a simple directed graph given the following.

- V: The set of nodes in the graph under consideration.
- E: The set of directed links (edges) in the graph.
- W: The number of wavelengths available on each link. The wavelengths are numbered from 1 to W.
- R: The set of connection requests. $M = |R|$. The requests are numbered from 1 to M. Each connection request has a source node and a destination node.[1] It requires a working lightpath and a backup lightpath from the stated source to the stated destination.
- G: The set of SRLGs. Each SRLG contains a set of links in E. Links in the same SRLG share the same risk, i.e., these links may break at the same time due to a destructive event. If two paths in the network have links in the same SRLG, we say that they are SRLG-joint. Otherwise, we call them SRLG-disjoint.

The goal is to determine a working lightpath and a backup lightpath for each connection request in R. The constraints are enumerated as follows.

C1: The number of wavelengths used on each link cannot not exceed W.

C2: The working lightpath and the backup lightpath, for each connection request, must be SRLG-disjoint.

C3: Two working lightpaths cannot use the same wavelength on the same link.

[1]Two different requests in C may have the same source and destination.

C4: A backup lightpath cannot share the same wavelength on the same link with any working lightpath.

C5: If two working lightpaths are SRLG-joint, their backup lightpaths cannot use the same wavelength on the same link.

The objective is to minimize the cost

$$\sum_{e \in E} (F_e + S_e) \tag{1}$$

where F_e is the number of wavelengths on link e used in working lightpaths, and S_e the number of wavelengths on link e used in backup lightpaths.

This problem has been studied and modeled as an integer linear programming (ILP) problem in [27]. It is an NP-complete problem [34]. An ILP approach involves too many constraints and variables, even for a small-sized network [35]. Therefore, dealing with any practical-sized network needs to resort to heuristics. Since the objective in this problem is to minimize the total number of wavelengths used in a routing scheme, a heuristic for it should have the following properties.

- The wavelength sharing among the backup lightpaths should be maximized.
- SRLG-disjointness among the working lightpaths should be encouraged so that backup lightpaths have a good chance of sharing wavelengths.
- The lightpaths should be as short as possible since a wavelength needs to be assigned to each link traversed by these lightpaths.

Compared to the extensive researches on dynamic protection where connection requests arrive one by one and future arrivals are not known (e.g., [36] and [37]), existing results on highly difficult offline SPPs, which handle all the connection requests to achieve global optimization objectives, are very limited. Specifically, [29] and [38] proposed to restrict the searching space of the ILP problem to a set of predefined alternative routes between each pair of nodes. The study in [39] applied the Lagrangean relaxation method to tackle the ILP formulations. Several network flow-based heuristics have also been proposed (e.g., [40]), where a good summary can be found in [41]. Other related yet different problems include the virtual topology design, where the wavelength of a connection can be changed on a limited set of intermediate nodes [42], [43], the length-limited hybrid protection problem [41], etc.

A heuristic for the SPP problem defined in this section has been proposed recently in [27]. To the best of our knowledge, it is the only practical heuristic for the problem in the literature. It first computes a working path and a backup path for each connection request. Then, it assigns a wavelength to each path. Finally, it rearranges the lightpaths to minimize the total number of the wavelengths used. It is easy to implement and have very low computational overheads. Its solution quality, although claimed to be suboptimal, can still be significantly improved as we will see later. Moreover, the heuristic of Zang–Ou–Mukherjee has no control parameters; therefore, it cannot be used in an indirect encoding EA.

One purpose of our study is to develop an indirect encoding EA using construction heuristics for producing solutions of high quality for complex optimization problems such as the SPP problem at modest computational costs.

III. Basic Heuristic (BH)

The proposed heuristic consists of three phases. In the first phase, a heuristic called the "working path router" ("Wrouter") routes a working path w_r for each connection request r in R. In the second phase, a heuristic called the "protection path router" ("Prouter") computes backup paths for all the working lightpaths established in the first stage. In the third phase, a heuristic called the "wavelength assigner" ("Wassigner") assigns a wavelength to each lightpath generated in the first two phases.

A. Wrouter

Let $P = \{p_1, p_2, \ldots, p_k\}$ be a set of paths in the graph. The SRLG-disjoint degree of path p_i in P, i.e., degree (p_i, P), is defined as the number of paths in P that are SRLG-disjoint from p_i. The SRLG-disjoint degree of P is defined as $\max_{p_i \in P}$ degree (p_i, P). Wrouter aims at establishing a set of working lightpaths of as large SRLG-disjoint degree as possible.

For each link e in the graph, its SRLG factor is defined as the number of SRLGs containing e. To increase the SRLG-disjoint degree of the set of working lightpaths, Wrouter discourages its working lightpaths from using links of high SRLG factors. The reason is that a path with links of high SRLG factor is very likely to be SRLG-joint with other paths.

Since working lightpaths cannot share the same wavelength on the same link and the number of wavelengths used on each link cannot exceed W, a single link, if possible, is prevented from being used by more than W lightpaths in Wrouter.

Wrouter routes working paths one by one in order of the connection requests $\pi = (\pi_1, \pi_2, \ldots, \pi_M)$. Suppose that for connection requests $\pi_1, \ldots, \pi_{k-1} (1 \leq k < M)$, a set of working lightpaths $\mathrm{WP}^{k-1} = \{\mathrm{wp}_{\pi_1}, \ldots, \mathrm{wp}_{\pi_{k-1}}\}$ has been established (in the case $k = 0$, no working lightpath has been established). If Wrouter failed in routing a working lightpath for request $r < k$, wp_r is set to be the empty path $\varnothing$. Wrouter routes the working lightpath wp_{π_k} for the connection request π_k in the following way.

Step 1) For each link in E, count the number of working lightpaths $\in \mathrm{WP}^{k-1}$ using it. Let $E_1 \subset E$ contain all the links that have been used by W lightpaths in WP^{k-1}.

Step 2) Compute the SRLG-disjoint degree of each nonempty working lightpath in WP^{k-1}. Let E_2 contain the links in $E \setminus E_1$ that are
  a) in the nonempty working lightpaths in WP^{k-1} with the smallest SRLG-disjoint degree;
  b) SRLG-joint with links specified in a).

Step 3) For each link $e \in E \setminus (E_1 \cup E_2)$, set its length to be

$$L(e) = (1 + c)^{\beta_e} \tag{2}$$

where β_e is its SRLG factor, and $c > 0$ is a control parameter.

For any link f in E_2, set its length to be

$$L(f) = \sum_{e \in E \setminus (E_1 \cup E_2)} L(e). \tag{3}$$

Step 4) Applying the Dijkstra algorithm [44] to the graph $(V, E \setminus E_1)$ with the link lengths defined in (2) and (3), compute the shortest path from the source to the destination of the connection request π_k as the working lightpath wp_{π_k}. If the Dijkstra algorithm fails in finding a path, set $\mathrm{wp}_{\pi_k} = \varnothing$.

The control parameters $\pi = (\pi_1, \pi_2, \ldots, \pi_M)$ and c can be regarded as the input to Wrouter, and the working lightpaths $\mathrm{WP} = \{\mathrm{wp}_1, \mathrm{wp}_2, \ldots, \mathrm{wp}_M\}$ are its output; therefore, Wrouter can be denoted by

$$\mathrm{WP} = \mathrm{Wrouter}(\pi, c). \tag{4}$$

Remark 1: If the lightpath wp_{π_k} uses any link in E_1, it will lead to an infeasible routing. This is why it is forbidden to use such links in wp_{π_k}. If the lightpath wp_{π_k} uses any link in E_2, the SRLG-disjoint degree of the resultant set of working lightpaths may be decreased. For this reason, Wrouter discourages wp_{π_k} from using any link in E_2. In fact, due to the link length settings in (2) and (3), wp_{π_k} will not use any link in E_2 if there is a path from the source node to the destination node of the request π_k in graph $(V, E \setminus (E_1 \cup E_2))$.

Remark 2: In (2), the higher the SRLG factor of a link is, the longer it is. In this way, Wrouter encourages wp_{π_k} to use links with low SRLG factors. Therefore, the SRLG-disjoint degree of the resultant set of working paths can be lowered.

Remark 3: For some control parameter settings, if there is no path from the source node to the destination node of the request π_k in the graph $(V, E \setminus E_1)$, then wp_{π_k} will be set to $\varnothing$ in Wrouter.

B. Prouter

Given a set of working lightpaths $\mathrm{WP} = \{\mathrm{wp}_1, \mathrm{wp}_2, \ldots, \mathrm{wp}_M\}$ for all the connection requests in R (where wp_r is for request r, and some wp_i in WP may be empty), the goal of Prouter is to route the backup lightpath bp_r for each connection request r.

In a similar way to Wrouter, Prouter establishes backup lightpaths one by one in order of the connection requests $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_M)$. Suppose that for connection requests $\sigma_1, \ldots, \sigma_{k-1} (1 \le k < M)$, a set of backup lightpaths $\mathrm{BP}^{k-1} = \{\mathrm{bp}_{\sigma_1}, \ldots, \mathrm{bp}_{\sigma_{k-1}}\}$ has been established (in the case $k = 0$, no backup lightpath has been established). Then Prouter routes the backup lightpath bp_{σ_k} for the connection request σ_k in the following way.

Step 1) If $\mathrm{wp}_{\sigma_k} = \varnothing$, set $\mathrm{bp}_{\sigma_k} = \varnothing$. Otherwise, go to Step 2).

Step 2) Let Q be the set containing all the backup lightpaths in BP^{k-1} whose corresponding working lightpaths are SRLG-joint with the working lightpath wp_{σ_k}. Let H contain links that
- a) have been used by W lightpaths in $Q \cup \mathrm{WP}$;
- b) have been used by the working lightpath wp_{σ_k};
- c) are SRLG-joint with links in the working path wp_{σ_k}.

For any link $e \in E \setminus H$, set its length to be

$$L(e) = 1. \tag{5}$$

Applying the Dijkstra algorithm to the graph $(V, E \setminus H)$ with the link lengths defined in (5), compute the shortest path from the source to the destination of the connection request σ_k as the backup lightpath bp_{σ_k}. If the Dijkstra algorithm fails in finding a path, set $\mathrm{bp}_{\sigma_k} = \varnothing$.

The control parameters $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_M)$ and the set of working lightpaths WP can be regarded as the input to Prouter, and the backup lightpaths $\mathrm{BP} = \{\mathrm{bp}_1, \mathrm{bp}_2, \ldots, \mathrm{bp}_M\}$ are its output. Therefore, Prouter can be denoted by

$$\mathrm{BP} = \mathrm{Prouter}(\sigma, \mathrm{WP}). \tag{6}$$

Remark 1: If the lightpath bp_{σ_k} uses any links in H defined in Step 3), it will lead to an infeasible routing. For this reason, Prouter prevents bp_{π_k} from using such links.

Remark 2: If there is no path from the source node to the destination node of the request σ_k in the graph $(V, E \setminus H)$ or wp_{σ_k}, Prouter will set $\mathrm{bp}_{\sigma_k} = \varnothing$.

C. Wassigner

Given the set of working lightpaths WP and the set of backup lightpaths BP for all the connection requests in R, the task of Wassigner is to assign wavelengths to every working lightpath and backup lightpath such that the total number of wavelengths used is minimized under the constraints listed as follows.

CW1: Two working lightpaths must be assigned different wavelengths if they traverse the same link.

CW2: A working path and a backup lightpath must be assigned different wavelengths if they traverse the same link.

CW3: Two backup lightpaths have to be assigned different wavelengths if they traverse the same link and their corresponding working lightpaths are SRLG-joint.

Viewing each lightpath as a node in a graph and each wavelength as a color, two nodes (lightpaths) are defined to be adjacent if they must be assigned different wavelengths. Then, the above problem becomes the well-known NP-hard graph coloring problem [45]. We use a first-fit heuristic for this problem. We assume that the number of wavelengths available are infinite and that the wavelengths are indexed by 1, 2, ... The heuristic, Wassigner, works as follows.

Step 1) Set $P = \mathrm{BP} \cup \mathrm{WP}$. Remove all the empty paths from P.

Step 2)

Step 2.1) Remove a lightpath p from P.

Step 2.2) Assign the allowable wavelength with the smallest index to p. A wavelength is allowable if assigning it to p does not violate CW1–CW3.

Step 3) If $P = \emptyset$, stop. Otherwise, go to Step 2).

There are several ways in Step 2.1) to select from P a lightpath to remove. In our implementation, we randomly pick a lightpath from P and remove it in Step 2.1). WP and BP are inputs to Wassigner. While the output is a wavelength assignment to each lightpath in $P = \text{BP} \cup \text{WP},\ A : P \rightarrow \{1, 2, \ldots\}$. Wassigner can be denoted by

$$A = \text{Wassigner}(\text{WP}, \text{BP}). \tag{7}$$

Remark 1: The wavelength assignment A generated in Wassigner will satisfy the constraints CW1–CW3. However, if the number of wavelengths used in A exceeds W, it may violate the constraint C1 (i.e., the number of wavelengths used on some paths may exceed W).

D. Structure of Heuristic Algorithm

For the SPP problem, the proposed BH has the control parameters $\pi = (\pi_1, \pi_2, \ldots, \pi_M)$, $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_M)$, and c. The output is a working lightpath set WP, a backup lightpath set BP, and a wavelength assignment $A : P \rightarrow \{1, 2, \ldots\}$. It works as

$$(\text{BP}, \text{WP}, A) = \text{BH}(\pi, \sigma, c)$$

where $\text{WP} = \text{Wrouter}(\pi, c)$, $\text{BP} = \text{Prouter}(\sigma, \text{WP})$, and $A = \text{Wassigner}(\text{WP}, \text{BP})$.

Any solution generated by BH will satisfy constraints C2–C5. An inappropriate setting of π, σ, and c may generate a solution that is incomplete (there are no working lightpaths or backup lightpaths for some requests) and/or violate constraint C1. It is not easy to develop a simple mathematical model for the relationship between a setting of algorithm parameters π, σ, and c and the quality of the solution constructed. Finding an optimal parameter setting turns out to be a black box optimization problem. An EA/G and a GA are used as tools for tuning the control parameters π and σ in our study.

IV. EA/G

Offspring generators play a crucial role in any EA. The proximate optimality principle [46], an underlying assumption in most (if not all) heuristics, assumes that good solutions have similar structure. This assumption is reasonable for most real-world problems. Based on this assumption, an ideal offspring assumption should be able to produce a solution that is close to the best solutions found so far. Only a few parent solutions are involved in crossover and mutation. A new solution generated by these two operators may be far from other best solutions found so far in the search. However, in EDAs [21], new solutions are sampled from a model that characterizes a promising area in the search space, but there is no mechanism in EDAs for directly controlling the similarity (often measured by distance) between new solutions and the best solutions found so far. Guided mutation combines global statistical information and location information of the best solutions found so far to overcome the shortcomings of GAs and EDAs. In the following, we present EA/G, which is used in our study for tuning the control parameters in the BH defined in the previous section.

A. Search Space and Objective Function

The search space in EA/G is Π, the set of all possible permutations of $I = \{1, 2, \ldots, M\}$. Each permutation has a cost. The task of EA/G is to find a permutation solution in Π with the (nearly) lowest cost.

B. Population and Probability Matrix

At each generation t, EA/G maintains a population of N solutions (i.e., permutations of I)

$$\text{Pop}(t) = \{\xi^1, \xi^2, \ldots, \xi^N\}$$

and a probability matrix

$$X(t) = \begin{pmatrix} x_{11}(t) & \cdots & x_{1M}(t) \\ \vdots & \ddots & \vdots \\ x_{M1}(t) & \cdots & x_{MM}(t) \end{pmatrix}$$

where $X(t)$ is the distribution of promising solutions in the search space. More precisely, $x_{ij}(t)$ is the probability that $\xi_i = j$ in a promising permutation solution $\xi = (\xi_1, \xi_2, \ldots, \xi_M)$.

C. Initialization

EA/G randomly chooses N permutations from Π as its initial population $P(0)$. The initial probability matrix $X(0) \in R^{M \times M}$ is set as

$$X(0) = \begin{pmatrix} \frac{1}{M} & \cdots & \frac{1}{M} \\ \vdots & \ddots & \vdots \\ \frac{1}{M} & \cdots & \frac{1}{M} \end{pmatrix}. \tag{8}$$

D. Update of Probability Matrix

Assume that the population at generation t is $\text{Pop}(t) = \{\xi^1, \xi^2, \ldots, \xi^N\}$, and the probability matrix at generation $t-1$ is $X(t-1)$. Then, the probability matrix $X(t) = (x_{ij}(t))_{M \times M}$ can be computed as

$$x_{ij}(t) = (1-\beta)\frac{1}{N}\sum_{k=1}^{N} I_{ij}(\xi^k) + \beta x_{ij}(t-1), \qquad (1 \le i, j \le M) \tag{9}$$

where

$$I_{ij}(\xi) = \begin{cases} 1, & \text{if } \xi(i) = j \\ 0, & \text{otherwise.} \end{cases}$$

$0 \le \beta \le 1$ is the learning rate. The bigger β is, the greater is the contribution of the solutions in $\text{Pop}(t)$ to the probability matrix $X(t)$.

E. Generation of New Solutions: Guided Mutation

Guided by the probability matrix $X = (x_{ij})_{M \times M}$, the guided mutation [25], [26] mutates an existing solution ξ to generate a new solution ζ. This operator also needs a control parameter $0 < \alpha < 1$. It works as follows:

$$\zeta = \text{Guided Mutation } (\xi, X, \alpha).$$

```
Input: A permutation $\xi = (\xi_1, \ldots, \xi_M)$, a probability matrix
    $X = (x_{ij})_{M\times M}$, and a positive parameter $\alpha < 1$.
Output: $\zeta = (\zeta_1, \ldots, \zeta_M)$, a permutation.
  Step 1) Randomly pick $[\alpha M]$ integers from $I = \{1, 2, \ldots, M\}$, and let these integers constitute a set $K \subset I$. Set $V = I \setminus K$ and $U = I \setminus \{\xi_l | l \in K\}$.
  Step 2) For each $i \in K$, set $\zeta_i = \xi_i$.
  Step 3) While $(U \neq \emptyset)$, uniformly randomly select $i$ from $V$, and then randomly draw $k$ from $U$ with probability
              $\frac{x_{ik}}{\sum_{j\in U} x_{ij}}$.
          Set $\zeta_i = k$, $U = U \setminus \{k\}$ and $V = V \setminus \{i\}$.
  Step 4) Return $\zeta$.
```

In the above guided mutation operator, ζ_i is directly copied from the parent ξ if $i \in K$. Otherwise, under the constraint that ζ is a permutation, it is randomly generated based on the probability matrix X. The larger α is, the more elements of ζ are directly copied from its parent ξ. In other words, α controls the similarity between the offspring and the parent.

In the conventional mutation for permutation vectors, several (often two) elements are randomly selected and then swapped. The probability that a permutation ζ being generated from parent ξ is entirely determined by the number of the positions where ζ and ξ are different. In contrast, the guided mutation mutates ξ based on the probability matrix X, which is learned and updated at each generation for modeling the distribution of promising solutions. It can be expected that offspring ζ fall in or close to a promising area in the search space. Meanwhile, randomness in Step 3) also provides diversity for the search.

F. Structure of EA/G

EA/G works as follows.

```
Step 0) Parameter Setting: Set the following control parameters:
          • $PopSize$: population size;
          • $NoNew$: number of new solutions generated at each generation;
          • $\alpha$: control parameter in Guided Mutation;
          • $\beta$: learning rate used in the update of the probability matrix;
          • $MaxGen$: maximum number of generations EA/G runs for.
Step 1) Initialization: Set $t := 0$. Initialize the probability matrix $X(t)$ by (8). Randomly generate $Popsize$ distinct permutations to form the initial population Pop(0). Let the best solution in Pop(0) be $\xi^*$.
Step 2) Guided Mutation: Independently apply the guided mutation operator to $\xi^*$ (i.e., $\xi^*, X(t)$, and $\alpha$ are inputs to the guided mutation) $NoNew$ times to generate $NoNew$ solutions.
Step 3) Selection: Select the $PopSize$ distinct best solutions from the solutions generated in Step 2) and Pop($t$). Let these selected solutions form Pop($t+1$). Let the best solution in Pop($t+1$) be $\xi^*$.
Step 4) Stopping Condition: If $t = MaxGen$, stop and output the best solution $\xi^*$; otherwise, set $t = t+1$.
Step 5) Update of Probability Matrix: Compute $X(t)$ by (9) and then go to Step 2).
```

In this algorithm, the members of the population Pop(t) are always distinct. This prevents $X(t)$ from becoming degenerate (i.e., all the elements in it are very close to 0 or 1) and thus provides diversity for the search. In Step 2), the guided mutation operator is always applied to the best solution ζ^*. Therefore, the new solutions generated are, to some extent, similar to the best solution found so far, which is desirable for a successful heuristic according to the proximate optimality principle [46]. In Step 3), the quality of a solution is measured by its cost. The smaller its cost is, the better it is.

V. GA

This section describes the GA used in our study for tuning the control parameters in BH. The search space is the permutation vector space.

```
Step 0) Parameter Setting: Set the following control parameters:
          • $PopSize$: population size;
          • $NoNew$: number of new solutions generated at each generation;
          • $\mu$ and $\gamma$: control parameters in mutation operator;
          • $MaxGen$: maximum number of generations the algorithm runs for.
Step 1) Initialization: Set $t := 0$. Randomly generate $Popsize$ distinct permutations to form the initial population Pop(0). Let the best solution in Pop(0) be $\xi^*$.
Step 2) Crossover and Mutation: Randomly select $NoNew$ pairs of permutations from Pop($t$) and perform the cycle crossover (CX) operator [8] on each pair to generate $NoNew$ new permutations. For each solution generated in Step 2), generate a uniform random number $rand$ in $(0, 1)$. If $rand < \mu$, mutate it by performing $[\gamma M]$ random swaps on it.
Step 3) Selection: Select the $PopSize$ distinct best solutions (i.e., with the smallest costs) from the solutions generated in Step 2) and Pop($t$). Let these selected solutions form Pop($t+1$).
Step 4) Stopping Condition: If $t = MaxGen$, stop and output the best solution $\xi^*$ in Pop($t+1$). Otherwise, set $t = t+1$, and go to Step 2).
```

Except for the mechanisms for generating new solutions and initialization, all the other components in the above GA are the same as in EA/G. The reason that we chose the CX operator is that this operator works well for other combination optimization problems [8].

TABLE I
PARAMETERS OF THE TEST NETWORKS

	No. of Nodes	No. of Links	No. of SRLGs	No. of Requests	No. of Wavelengths Available
Setting 1	19	62	31	100	32
Setting 2	19	62	29	100	32
Setting 3	19	62	27	100	32
Setting 4	24	86	43	100	32
Setting 5	24	86	41	100	32
Setting 6	24	86	40	100	32
Setting 7	24	86	39	100	32
Setting 8	31	94	47	150	64
Setting 9	50	200	85	250	64
Setting 10	50	200	85	275	64
Setting 11	50	200	85	300	64
Setting 12	60	200	90	250	64
Setting 13	65	200	100	250	64
Setting 14	65	200	100	300	64
Setting 15	100	340	150	350	64

VI. TUNING THE PARAMETER SETTING OF HEURISTIC: BH/EA/G AND BH/GA

The proposed heuristic BH may generate an incomplete solution or a solution violating constraint C1. We need to measure its performance in tuning the control parameters in BH. For a solution $(\text{BP}, \text{WP}, A)$ generated by BH, we use the following function as its cost function:

$$2M|E|(N_1 + N_2) + \sum_{e\in E}(F_e + S_e) \tag{10}$$

where F_e is the number of wavelengths on link e used in working lightpaths, and S_e the number of wavelengths on link e used in backup lightpaths. M is the number of the connection requests, $|E|$ is the number of all the links in E, N_1 is the number of empty lightpaths in BP $\cup$ WP, and

$$N_2 = \begin{cases} 0, & \text{if } W \geq K \\ K - W, & \text{otherwise} \end{cases}$$

where K is the number of wavelengths used in A.

Obviously, due to the penalty term in (10), the cost of an infeasible or incomplete solution is always higher than that of a feasible solution.

For each parameter setting of (π, σ, c) in BH, we define its cost $\text{cost}(\pi, \sigma, c)$ to be the cost of the solution generated by BH with such a parameter setting. The proposed procedure for tuning these parameters works as follows.

BH/EA/G (BH/GA)

Step 1) Tuning c.

Step 1.1) Randomly generate ten pairs of permutations $(\pi^1, \sigma^1), (\pi^2, \sigma^2), \ldots, (\pi^{10}, \sigma^{10})$. Let $c_i = (i/10)(i = 1, 2, \ldots, 10)$.

Step 1.2) Compute

$$c^\star = \arg \min_{c\in\{c_1,c_2,\ldots,c_{10}\}} \frac{1}{10}\sum_{j=1}^{10} \text{cost}(\pi^j, \sigma^j, c).$$

Step 2) Tuning π.

Step 2.1) Randomly generate a permutation $\tilde{\sigma}$.

Step 2.2) Use EA/G (GA) to tune π, where the cost of π is set as $\text{cost}(\pi, \tilde{\sigma}, c^\star)$. Set $\pi^\star$ to be the best setting of π found in EA/G (GA).

Step 3) Tuning σ.
Use EA/G (GA) to tune σ, where the cost of σ is set as $\text{cost}(\pi^\star, \sigma, c^\star)$. Set $\sigma^\star$ to be the best setting of σ found in EA/G (GA).

Then, the best solution found is the one generated by BH with parameter setting $(\pi^\star, \sigma^\star, c^\star)$.

We can iterate the above procedure many times in order to lower the cost of the solution obtained as in the alternating variable optimization method [47]. Taking the computational overhead into consideration, we chose not to perform the iteration in our experimental study.

VII. EXPERIMENTAL RESULTS

A. Test Networks

The basic characteristics of the test network instances are listed in Table I.

For each network parameter setting, we test two network examples. In the first one, the links in the same SRLG are adjacent. In the second one, the links in each SRLG are randomly selected from E. The first test example with parameter setting a is denoted by $T1 - a$, and the second test example is denoted by $T2 - a$.

A test network example is generated as follows.

1) Let $G = (V, E)$. E has N_L distinct links that are randomly selected.
2) Randomly divide all the links in E into $|G|$ groups. Each group is an SRLG. In the case of test example $T1 - a$, the links in the same SRLG should be adjacent. In other words, let H be the subgraph induced by SRLG, then H should be connected if the directions of the links are not considered.
3) Randomly select M connection requests. Let the number of wavelengths available on each link be W.
4) Apply the heuristic of Zang–Ou–Mukherjee [27] to the generated problem. If a feasible solution can be found, then stop. This problem will be taken as a test network instance. Otherwise, go to Step 1).

The above procedure can produce an instance of the SPP problem that is guaranteed to have at least one feasible solution. As pointed out in [27], linear programming methods are not suitable for dealing with networks as large as the above networks.

B. Parameter Setting in EA/G and GA in Comparison

In our experimental study, we set $NoNew = 100$ and $MaxGen = 100$ in both EA/G and GA. Therefore, both BH/EA/G and BH/GA need to call BH 20 000 times [10 000 times for tuning π in Step 2) of BH/EA/G (BH/GA) and 10 000 times for tuning σ].

We set $Popsize = 50$ in EA/G.

In EA/G, there are another two parameters, i.e., α (in the guided mutation operator) and β (used in updating the probability matrix), while GA has three more parameters to set, i.e., μ, γ used in the mutation operator, and $PopSize$ used for controlling the selection pressure. To determine an appropriate setting for these parameters, we have run BH/EA/G and BH/GA on $T1-1$ for $\alpha, \beta = 0.0, 0.1, \ldots, 0.9$, and $\mu, \gamma = 0.0, 0.1, \ldots, 0.9$, $Popsize = 20, 30, 40, 50, 60, 70, 80$, respectively. We have found that $\alpha = 0.1$ and $\beta = 0.2$ is the best setting for EA/G and $\mu = 0.1$, $\gamma = 0.3$, and $Popsize = 40$ is the best for GA for $T1-1$. In the following comparison study, we always use these settings for EA/G and GA, although it does not mean that these settings are the best for all the test problems.

C. Comparison Results

We have compared the following four algorithms in our experimental study:

1) BH/EA/G;
2) BH/GA;
3) BH/R, which has the same structure as BH/EA/G and BH/GA, except that it tests 10 000 random permutations for π in Step 2) and 10 000 random permutations for σ in Step 3), respectively;
4) ZOM-H, heuristic of Zang–Ou–Mukherjee.

The first three algorithms need to call BH the same number of times in each run. Due to the computational cost, each BH/EA/G, BH/GA, and BH/R has been run independently for ten times on each test network instance.

Tables II and III shows the experimental results including the following:

- *time*: The average run time (in seconds) of each algorithm for each test instance.
- *cost*: The lowest cost found in the heuristic of Zang–Ou–Mukherjee. Since there is no randomness in this heuristic, we only run it once for obtaining its $cost$.
- *best*: The lowest cost found in ten independent runs for each test instance by BH/EA/G, BH/GA, and BH/R, respectively.
- *avg*: The average of the solution costs in ten independent runs for BH/EA/G, BH/GA, and BH/R, respectively.
- *std*: The standard derivation of the lowest costs found in ten independent runs for BH/EA/G, BH/GA, and BH/R, respectively.

All the experiments were performed on a cluster of Athlon MP 1900s (1.6 GHz). It is from Table II that the running time for the heuristic of Zang–Ou–Mukherjee is from a few seconds to about 4 min for the test network instances, while the other three algorithms need from 2 min to 4 h for each run. For offline applications that typically plan the allocations of connection requests for a long period of time (days or even months), a 4-h computation time is totally acceptable.

TABLE II
AVERAGE RUN TIME (IN SECONDS) OF HA, BH/R, BH/GA, AND BH/EA/G ON TEST NETWORK INSTANCES

instances	ZOM-H	BH/R	BH/GA	BH/EA/G
T_1-1	2.51	76.40	101.80	141.80
T_1-2	2.74	77.10	99.50	158.50
T_1-3	3.10	80.60	139.50	161.30
T_1-4	2.13	82.80	129.60	149.10
T_1-5	2.01	77.50	114.40	136.70
T_1-6	3.10	77.60	141.20	159.60
T_1-7	2.32	70.40	135.30	127.60
T_1-8	41.10	675.10	874.10	1180.30
T_1-9	63.30	1586.30	2005.00	2601.70
T_1-10	150.60	2115.70	3133.00	3492.50
T_1-11	161.60	2010.90	3512.70	3954.20
T_1-12	165.40	2174.60	3599.10	3741.25
T_1-13	200.10	2875.10	4112.80	4385.90
T_1-14	241.10	8081.90	12091.53	14951.50
T_1-15	246.60	7651.40	12109.51	13388.40
T_2-1	2.55	82.10	127.50	151.30
T_2-2	2.69	89.60	147.20	161.40
T_2-3	2.05	71.30	128.70	139.60
T_2-4	2.96	97.80	168.10	187.40
T_2-5	2.29	85.10	120.60	151.40
T_2-6	2.67	79.80	154.50	179.50
T_2-7	2.81	75.10	142.50	168.70
T_2-8	46.40	689.50	671.70	848.40
T_2-9	67.90	1480.60	2000.10	1861.50
T_2-10	152.80	2061.50	2975.80	3364.90
T_2-11	164.20	2513.60	3552.90	3998.90
T_2-12	163.50	2315.50	3132.50	3776.20
T_2-13	210.60	2910.60	4010.90	4408.80
T_2-14	251.70	7916.90	11257.90	14225.40
T_2-15	239.80	7412.30	11219.40	13395.90

Table III clearly shows that BH/EA/G, BH/GA, and BH/R are much better than the heuristic of Zang–Ou–Mukherjee in terms of solution quality.

Table III also shows that *std* are the about the same for BH/EA/G, BH/GA, and BH/R for the test instances, and BH/EA/G and BH/GA perform significantly better than BH/R for all the test problems with the same number of BH calls in terms of *best* and *avg*. These results imply that EA's search mechanism is very effective in tuning the control parameters in BH. Perhaps this is because the proximate optimality principle also holds for the BH control parameters as in many other real-world problems.

It is evident from Table III that BH/EA/G performs better than BH/GA in all the test instances but $T_1 - 7$. These results suggest that the combination of global statistical information and location information in the guided mutation operator does improve the performance of the search.

Figs. 1 and 2 give the evolution of the average cost of the best solutions found in ten runs with the number of BH calls in tuning π and σ on two test instances. The first 10 000 BH calls are for tuning π, while the last 10 000 calls are for tuning σ. It is very clear from these four figures that there are significant

TABLE III
SOLUTION QUALITY OF HA, BH/R, BH/GA, AND BH/EA/G ON TEST NETWORK INSTANCES

	ZOM-H	BH/R			BH/GA			BH/EA/G		
instances	cost	*best*	*avg*	*std.*	*best*	*avg*	*std.*	*best*	*avg*	*std.*
T_1-1	478.00	404.00	409.60	4.81	369.00	381.50	4.18	365.00	372.50	4.03
T_1-2	458.00	401.00	403.80	3.06	378.00	382.70	5.06	368.00	374.55	4.57
T_1-3	487.00	405.00	410.90	5.60	370.00	381.40	7.38	364.00	368.70	6.01
T_1-4	445.00	411.00	420.05	6.94	394.00	401.50	3.05	377.00	384.60	5.58
T_1-5	492.00	427.00	430.90	5.23	393.00	410.30	4.56	384.00	390.75	5.05
T_1-6	462.00	417.00	420.20	3.38	390.00	399.00	7.49	377.00	384.90	5.72
T_1-7	451.00	420.00	423.30	3.83	383.00	386.40	2.47	384.00	403.10	6.18
T_1-8	785.00	701.00	710.25	7.78	660.00	671.40	8.07	636.00	649.20	4.86
T_1-9	1389.00	1308.00	1314.15	5.41	1257.00	1268.40	8.10	1221.00	1239.10	0.00
T_1-10	1516.00	1467.00	1474.30	8.35	1407.00	1432.20	10.00	1348.00	1372.50	0.00
T_1-11	1612.00	1514.00	1519.20	7.66	1453.00	1467.50	7.64	1426.00	1439.20	0.00
T_1-12	1661.00	1494.00	1502.40	9.46	1447.00	1461.10	7.99	1414.00	1424.80	0.00
T_1-13	1834.00	1621.00	1636.40	5.19	1557.00	1581.90	10.76	1526.00	1551.60	0.00
T_1-14	2169.00	1924.00	1931.20	8.34	1829.00	1857.10	11.23	1809.00	1825.90	0.00
T_1-15	2511.00	2298.00	2328.70	10.57	2208.00	2253.70	12.57	2195.00	2215.00	0.00
T_2-1	452.00	392.00	398.75	5.41	367.00	371.70	3.02	355.00	364.70	3.33
T_2-2	458.00	383.00	394.25	10.50	364.00	376.50	3.77	355.00	363.45	3.53
T_2-3	501.00	436.00	440.05	11.10	410.00	419.60	6.50	399.00	404.50	5.13
T_2-4	479.00	433.00	440.10	8.60	410.00	417.40	6.51	399.00	406.30	4.15
T_2-5	444.00	405.00	408.60	7.50	375.00	390.50	3.24	366.00	376.35	6.13
T_2-6	467.00	422.00	426.65	9.00	396.00	405.90	4.76	380.00	388.35	3.26
T_2-7	540.00	462.00	467.25	5.10	435.00	444.90	5.93	419.00	428.90	4.85
T_2-8	898.00	762.00	776.75	10.50	736.00	741.50	5.01	715.00	727.80	6.72
T_2-9	1369.00	1264.00	1277.35	15.60	1223.00	1239.10	8.41	1192.00	1203.35	8.63
T_2-10	1415.00	1376.00	1383.10	15.80	1330.00	1353.20	7.34	1213.00	1225.70	7.57
T_2-11	1569.00	1518.00	1527.60	9.63	1475.00	1500.90	11.37	1397.00	1401.80	9.10
T_2-12	1649.00	1554.00	1560.00	7.89	1491.00	1501.40	12.53	1451.00	1476.90	11.41
T_2-13	1785.00	1654.00	1666.00	10.70	1609.00	1615.50	6.76	1567.00	1582.70	8.75
T_2-14	2114.00	1970.00	1984.40	15.10	1904.00	1917.70	13.60	1866.00	1889.70	10.50
T_2-15	2446.00	2265.00	2282.70	11.30	2198.00	2221.90	14.90	2155.00	2183.30	9.81

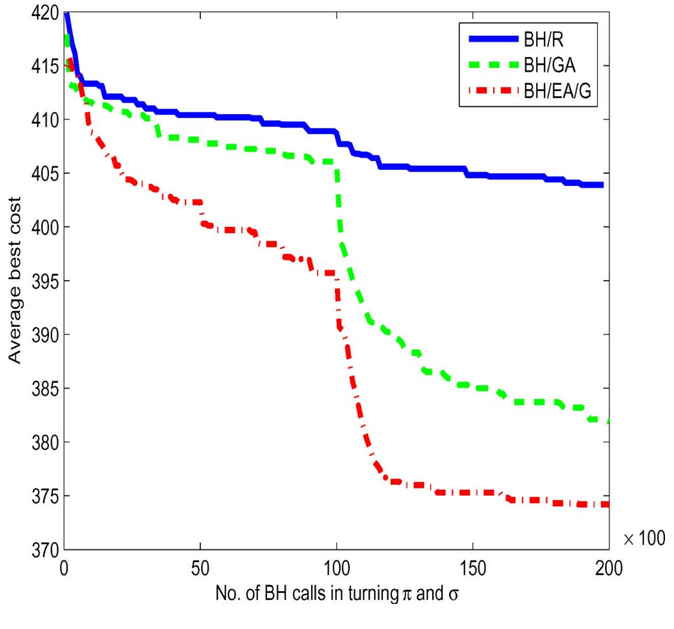

Fig. 1. Evolution of the average of the lowest solution costs with the number of BH calls in tuning π and σ in BH/R, BH/GA, and BH/EA/G for $T_1 - 2$.

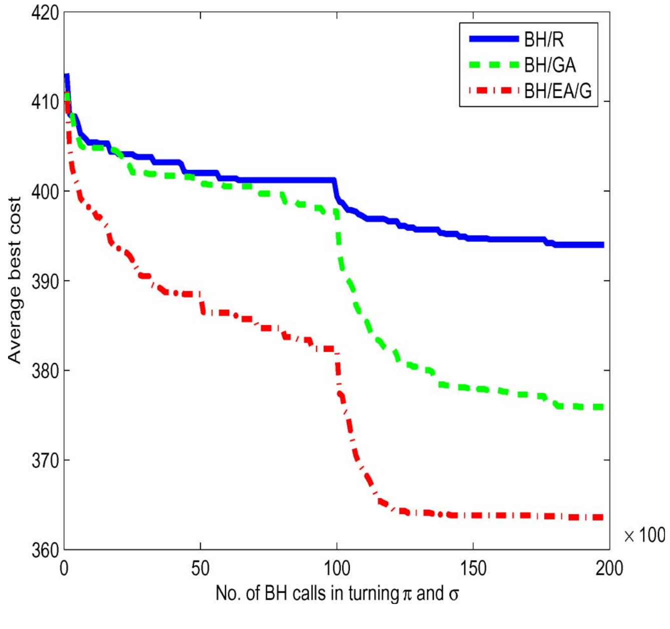

Fig. 2. Evolution of the average of the lowest solution costs with the number of BH calls in tuning π and σ in BH/R, BH/GA, and BH/EA/G for $T_2 - 2$.

decreases in the solution costs in both phases of tuning π and σ in BH/EA/G and BH/GA.

The test problems and the C++ code of all the algorithms in this paper can be found in cswww.essex.ac.uk/staff/zhang/.

VIII. CONCLUSION

Some real-world optimization problems may have complex data structures. Indirect encoding EAs using construction heuristics represent a feasible approach of dealing with such problems. In this paper, such an approach has been applied for solving the SPP problem in WDM networks with SRLGs. The proposed method optimizes the control parameters of a basic construction heuristic for the problem by using EA/G. The construction heuristic BH has three phases with three different parameters, which largely determine the performance of BH. Two parameters are permutations, and the third one is real valued. These three parameters are tuned separately to search for a nearly optimal solution to the problem. Our experimental study shows that the approach performs much better than the heuristic of Zang–Ou–Mukherjee. More significantly, by comparing the performances of EA/G, GA, and a pure random method for tuning the control parameters, we have shown that EAs are suitable for such a task, and the guided mutation operator does improve the performance of EAs.

In the future, we intend to apply this approach to solve other hard search and optimization problems.

Acknowledgment

The authors would like to thank for H. Zang for helpful discussions on the heuristic of Zang–Ou–Mukherjee and the SPP problem and J. Ford and G. Boshnakov for constructive comments.

References

[1] D. Corne, M. Dorigo, and F. Glover, Eds., *New Ideas in Optimisation*, New York: McGraw-Hill, 1999.
[2] H. E. William, E. N. Krasnogor, and J. E. Smith, Eds., *Recent Advances in Memetic Algorithms*, New York: Springer-Verlag, 2005.
[3] P. Moscato and C. Cotta, "A gentle introduction to memetic algorithms," in *Handbook of Meta-Heuristics*, F. Glover and G. Kochenberger, Eds. Norwell, MA: Kluwer, 2003, pp. 105–144.
[4] N. Krasnogor, "Studies on the theory and design space of memetic algorithms," Ph.D. dissertation, Univ. West England, Bristol, U.K., 2002.
[5] Y. S. Ong and A. J. Keane, "Meta-Lamarckian in memetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.
[6] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 392–403, Aug. 1998.
[7] Q. Zhang and Y.-W. Leung, "Orthogonal genetic algorithm for multimedia multicast routing," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 53–62, Apr. 1999.
[8] P. Merz and B. Freisleben, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Trans. Evol. Comput.*, vol. 4, no. 4, pp. 337–352, Nov. 2000.
[9] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 141–152, Feb. 2006.
[10] H.-L. Fang, P. M. Ross, and D. Corne, "A promising hybrid GA/heuristic approach for open-shop scheduling problems," in *Proc. 11th ECAI*, A. Cohn, Ed., 1994, pp. 590–594.
[11] E. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of Meta-Heuristics*, F. Glover and G. Kochenberger, Eds. Norwell, MA: Kluwer, 2003, pp. 457–474.
[12] N. Krasnogor and J. E. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy and design issues," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 474–488, Oct. 2005.
[13] B. A. Huberman, R. M. Lukose, and T. Hogg, "An economics approach to hard computational problems," *Science*, vol. 275, no. 5296, pp. 51–54, Jan. 3, 1997.
[14] A. Kapsalis, V. J. Rayward-Smith, and G. D. Smith, "Solving the graphical Steiner tree problem using genetic algorithms," *J. Oper. Res. Soc.*, vol. 44, no. 4, pp. 397–406, 1993.
[15] Y. Xiong, B. Golden, and E. Wasil, "A one-parameter genetic algorithm for the minimum labeling spanning tree problem," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 55–60, Feb. 2005.
[16] D. A. Pelta and N. Krasnogor, "Multimeme algorithms using fuzzy logic based memes for protein structure prediction," in *Recent Advances in Memetic Algorithms*. New York: Springer-Verlag, 2005, pp. 49–64.
[17] P. E. Hotz, "Comparing direct and developmental encoding schemes in artificial evolution: A case study in evolving lens shapes," in *Proc. IEEE Congr. Evol.*, 2004, pp. 752–757.
[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[19] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: Univ. of Michigan Press, 1975.
[20] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1996.
[21] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA: Kluwer, 2002.
[22] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-94-163, 1994.
[23] Q. Zhang and H. Muehlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 127–136, Apr. 2004.
[24] Q. Zhang, "On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 80–93, Feb. 2004.
[25] Q. Zhang, J. Sun, and E. P. K. Tsang, "Evolutionary algorithm with the guided mutation for the maximum clique problem," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 192–200, Apr. 2005.
[26] Q. Zhang, J. Sun, E. P. K. Tsang, and J. A. Ford, "Combination of guided local search and estimation of distribution algorithm for solving quadratic assignment problem," in *Proc. Bird Feather Workshops, Genetic and Evol. Comput. Conf.*, 2004, pp. 42–48.
[27] H. Zang, C. Ou, and B. Mukherjee, "Path-backup routing and wavelength-assignment (RWA) in WDM mesh networks under duct-layer constraints," *IEEE/ACM Trans. Netw.*, vol. 11, no. 2, pp. 248–258, Apr. 2003.
[28] B. Mukherjee, *Optical Communication Networks*. New York: McGraw-Hill, 1997.
[29] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh network, Part I—Protection," in *Proc. IEEE Infocom*, 1999, pp. 744–751.
[30] C. Ou, J. Zhang, H. Zang, L. H. Sahasrabuddhe, and B. Mukherjee, "New and improved approaches for shared-path protection in WDM mesh networks," *J. Lightw. Technol.*, vol. 22, no. 5, pp. 1223–1232, May 2004.
[31] Y. Xiong, D. Xu, and C. Qiao, "Achieving fast and bandwidth efficient shared-path protection," *J. Lightw. Technol.*, vol. 21, no. 2, pp. 365–371, Feb. 2003.
[32] J. Strand, A. Chiu, and R. Tkach, "Issues for routing in the optical layer," *IEEE Commun. Mag.*, vol. 39, no. 2, pp. 81–87, Feb. 2001.
[33] P. Sebos, J. Yates, A. Greenberg, and D. Rubenstein, "Effectiveness of shared risk link group auto-discovery in optical networks," in *Proc. Opt. Fiber Commun. Conf.*, Mar. 2002, pp. 493–495.
[34] J. Q. Hu, "Diverse routing in optical mesh networks," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 489–494, Mar. 2003.
[35] H. Zang, J. P. Jue, and B. Mukherjeee, "A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks," *Opt. Netw. Mag.*, vol. 1, no. 1, pp. 47–60, Jan. 2000.
[36] S. Sengupta and R. Ramamurthy, "From network design to dynamic provisioning and restoration in optical cross-connect mesh networks: An architectural and algorithmic overview," *IEEE Netw.*, vol. 15, no. 4, pp. 46–54, Jul./Aug. 2001.
[37] G. Ellinas, E. Bouillet, R. Ramamurthy, J. Labourdette, S. Chauduri, and K. Bala, "Routing an restoration architectures in mesh optical networks," *SPIE Opt. Netw. Mag.*, vol. 4, no. 1, pp. 91–106, Jan./Feb. 2003.
[38] M. Sridharan, A. Somani, and M. Salapaka, "Approaches for capacity and revenue optimization in survivable WDM networks," *J. High Speed Netw.*, vol. 10, no. 2, pp. 109–124, 2001.
[39] D. Doshi *et al.*, "Optical network design and restoration," *Bell Labs Tech. J.*, vol. 4, no. 1, pp. 58–84, 1999.
[40] P. Ho and H. Mouftah, "Reconfigurations of spare capacity for MPLS-based recovery for the Internet backbone networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 73–84, Feb. 2004.
[41] L. Shen, X. Yang, and B. Ramamurthy, "Shared risk link group (SRLG)-diverse path provisioning under hybrid service level agreement in wavelength-routed optical mesh networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 4, pp. 918–931, Aug. 2005.
[42] P. Datta, M. T. Frederick, and A. K. Somani, "Sub-graph routing: A novel fault-tolerant architecture for shared-risk link group failures in WDM optical networks," in *Proc. DRCN*, 2003, pp. 296–303.
[43] A. Todimala and B. Ramamurthy, "Survivable virtual topology routing under shared risk link groups in WDM networks," in *Proc. BoardNets*, 2004, pp. 130–139.
[44] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijiver, *Combinatorial Optimization*. Hoboken, NJ: Wiley, 1998.
[45] T. R. Jensen and B. Toft, *Graph Coloring Problems*. New York: Wiley Interscience, 1995.
[46] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA: Kluwer, 1998.
[47] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Hoboken, NJ: Wiley, 1987.

Qingfu Zhang (M'01) received the B.Sc. degree in mathematics from Shanxi University, Shanxi, China, in 1984, and the M.Sc. degree in applied mathematics and the Ph.D. degree in information engineering from Xidian University, Xi'an, China, in 1991 and 1994, respectively.

From 1994 to 2000, he was with the National Laboratory of Parallel Processing and Computing, National University of Defence Science and Technology, China; the Hong Kong Polytechnic University, Hong Kong; the German National Research Center for Information Technology, Fraunhofer-Gesellschaft, Germany; and the University of Manchester Institute of Science and Technology, U.K. He is currently a Reader with the Department of Computer Science, University of Essex, Colchester, U.K. His main research areas are evolutionary computation, optimization, neural networks, data analysis, and their applications.

Dr. Zhang is an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B and a Guest Editor of a forthcoming special issue on evolutionary algorithms based on probabilistic models in the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

Jianyong Sun received the B.Sc. degree in mathematics from Xi'an Jiaotong University, Xi'an, China, in 1997, and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2005.

In 2000, he was a Research Assistant with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. From 2002 to 2003, he was a Senior Research Officer with the Department of Computer Science, University of Essex. He is currently a Research Associate with the School of Computer Science, University of Birmingham, Birmingham, U.K. His main research areas are evolutionary computation, optimization, theory and algorithms of metaheuristics, telecommunication networks, and machine learning.

Gaoxi Xiao (M'00) received the B.Sc. and M.Sc. degrees in applied mathematics from Xidian University, Xi'an, China, in 1991 and 1994, respectively, and the Ph.D. degree from the Hong Kong Polytechnic University, Kowloon, Hong Kong, in 1999.

From 1994 to 1995, he was with the Institute of Antenna and Electromagnetic Scattering, Xidian University. In 1999, he was a Postdoctoral Fellow with the Department of Electronic Engineering, Polytechnic University, Brooklyn, NY. In 1999–2001, he was a Visiting Scientist with the Center for Advanced Telecommunications Systems and Services, University of Texas, Dallas. Since October 2001, he has been an Assistant Professor with the Division of Communication Engineering, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include optical and wireless networking, complex networks, algorithm design and analysis, and network security.

Edward Tsang (M'04) received the B.S. degree in business administration from the Chinese University of Hong Kong, Hong Kong, in 1977 and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 1987.

He has broad interest in applied artificial intelligence, particularly in computational finance, scheduling, heuristic search, constraint satisfaction, and optimization. He is currently a Professor in computer science with the University of Essex, Colchester, U.K., where he leads the Computational Finance Group and the Constraint Satisfaction and Optimization Group. He is also the Deputy Director of the Centre for Computational Finance and Economic Agents, an interdisciplinary center.

Dr. Tsang chaired the Technical Committee for Computational Finance under the IEEE Computational Intelligence Society in 2004 and 2005.

Evolutionary Algorithms Refining a Heuristic: A Hybrid Method for Shared-Path Protections in WDM Networks Under SRLG Constraints

Qingfu Zhang, *Member, IEEE*, Jianyong Sun, Gaoxi Xiao, *Member, IEEE*, and Edward Tsang, *Member, IEEE*

***Abstract*—An evolutionary algorithm (EA) can be used to tune the control parameters of a construction heuristic to an optimization problem and generate a nearly optimal solution. This approach is in the spirit of indirect encoding EAs. Its performance relies on both the heuristic and the EA. This paper proposes a three-phase parameterized construction heuristic for the shared-path protection problem in wavelength division multiplexing networks with shared-risk link group constraints and applies an EA for optimizing the control parameters of the proposed heuristics. The experimental results show that the proposed approach is effective on all the tested network instances. It was also demonstrated that an EA with guided mutation performs better than a conventional genetic algorithm for tuning the control parameters, which indicates that a combination of global statistical information extracted from the previous search and location information of the best solutions found so far could improve the performance of an algorithm.**

***Index Terms*—Estimation of distribution algorithms (EDAs), evolutionary algorithm (EA), guided mutation, hyperheuristics, memetic algorithm (MA), network protection, shared-risk link group (SRLG).**

I. Introduction

THE COMBINATION of evolutionary algorithms (EAs) and problem-specific heuristics has been proven very successful in dealing with hard search and optimization problems. The majority of current combination schemes adopt one or more of the following three related approaches.

1) Memetic algorithm (MA) [1]–[8]: MAs are inspired by cultural evolution. They employ one or several problem-specific heuristics to improve and/or repair some or all of the solutions generated by EA offspring generators (such as crossover and mutation). In a canonical MA, a single heuristic local search procedure is applied to every newly generated solution, while adaptive MAs use a number of heuristics, and the decision on which heuristic to improve a new solution is made dynamically. Therefore, adaptive MAs allow both cooperation and competition among different heuristics. A very recent classification of adaptive MAs based on adaptation level can be found in [9].
2) Hyperheuristic [10]–[12]: This is a general methodology in search and optimization. Typically, a hyperheuristic does not directly work on the solution space of the optimization problem. Instead, it manages a set of knowledge-poor and low-level heuristics that can modify or repair an existing solution to the problem. At any given decision point, high-level heuristics choose a low-level heuristic and then apply it to the solutions obtained from the previous stage of search. A hyperheuristic can use an EA as its high-level heuristic for searching good low-level heuristics. Portfolio algorithms, based on ideas from economics, also aim at combining different heuristics for solving hard optimization problems [13].
3) Indirect encoding EAs using construction heuristics [5], [14]–[17]: Suppose that we have a problem-specific construction heuristic for an optimization problem. The heuristic has a set of control parameters with relatively simple data structures. For any given parameter setting, the heuristic can construct a solution to the problem, the quality of which entirely depends on the parameter setting. An EA can be used to tune these control parameters to find a nearly optimal parameter setting and thus generate a good solution to the original problem. This approach can be regarded as an instance of indirect encoding EAs since the actual search space of the EA is the parameter space of the heuristic and each parameter setting can be decoded (i.e., transformed) to a solution to the original problem via the heuristic. If solutions to the problem have a complex data structure, and a parameterized construction heuristic is relatively easy to design, this approach could be a reasonable choice.

Conventional EAs [18]–[20] mainly employ crossover and mutation for generating new solutions. Generally, only a few (often two in crossover and one in mutation) parent solutions are directly involved in these operators. There is no mechanism in conventional EAs for extracting global statistical information from the previous search and using it for guiding the further search. Estimation of distribution algorithms (EDAs) [21]–[24] work in a quite different way: they maintain a probability model for characterizing the distribution of promising solutions at each generation. The model is updated based on the global

Manuscript received April 26, 2005; revised October 10, 2005 and February 6, 2006. This paper was recommended by Guest Editor Y. S. Ong.

Q. Zhang and E. Tsang are with the Department of Computer Science, University of Essex, CO4 3SQ Colchester, U.K.

J. Sun is with the School of Computer Science, University of Birmingham, B15 2TT Birmingham, U.K.

G. Xiao is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2006.883269

statistical information extracted from the current population. New solutions are generated by sampling from this model. However, information about the location of the best solutions found so far has not been directly utilized. Recently, we have proposed a new operator called guided mutation [25], [26] for generating new solutions in EAs. Guided mutation combines the global statistical information and location information of the solutions found so far so that the shortcomings of EAs and EDAs are efficiently overcome.

In this paper, an indirect encoding EA using a construction heuristic is proposed for the shared-path protection (SPP) problem in wavelength division multiplexing (WDM) optical networks under shared-risk link group (SRLG) constraints [27]. This problem requires finding a working lightpath and a backup lightpath for each of the set of connection requests with the objective of minimizing the wavelength capacities consumed in these lightpaths. Each feasible solution to this problem, therefore, consists of a set of working lightpaths and a set of backup lightpaths subject to a large number of constraints. If we encode a solution as a binary string, the resultant problem size, for a practical-sized network, would become prohibitively large to deal with [27]. It is also not an easy task to design EA operators such as crossover and mutation for operating directly on these solutions. Therefore, the commonly used framework of MAs is not very suitable for this problem. In our proposed method for this problem, a three-phase parameterized construction heuristic is used to construct a solution. The first and second phases are for constructing working lightpaths and backup lightpaths, respectively. The third phase assigns a wavelength to each lightpath. The quality of the solution generated in the construction heuristic is determined by its control parameters: two permutation vectors π and σ, and a real parameter c. An EA with guided mutation (EA/G) and a genetic algorithm (GA) are employed for optimizing π and σ, while c is tuned by testing its several representative values. The experimental results show that EA/G outperforms the conventional GA in tuning these control parameters, which indicates that the combination of location information and global statistical information can improve the performance of an EA. The results also show that our proposed approach performs significantly better than the heuristic of Zang–Ou–Mukherjee [27].

This paper is organized as follows. Section II describes in detail the SPP problem in WDM optical networks under SRLG constraints. Section III presents the construction heuristic for the SPP problem. Sections IV and V introduce GA and EA/G, which are used for tuning the control parameters. The framework of the proposed approach for the SPP problem is given in Section VI. Section VII presents the experimental results. Finally, Section VIII concludes this paper.

II. Problem Definition

In this paper, we consider the network protection problem in WDM networks. Due to their capability of efficiently utilizing the huge bandwidth of optical fibers, WDM networks are now the mainstream solution for supporting high-speed long-distance communications. In such high-speed networks, however, a single link failure can lead to serious service interruptions. Therefore, network protection is of critical importance. Among the most popular protection methods is the so-called path protection [28], where we set up two link-disjoint lightpaths (named as the working lightpath and the backup lightpath, respectively) between each source–destination pair. Here, a lightpath is a directed path from source to destination along which all the links use the same wavelength. To save on network capacity, different backup lightpaths can share the same wavelength channel as long as their respective working lightpaths are not going through any common link. Extensive researches have been carried out on investigating this problem (e.g., [29]–[31]).

One of the most important recent developments in WDM network protection is the introduction of the SRLG concept [32], [33]. SRLG is defined as a set of network components with a significant probability of failing simultaneously, e.g., fibers going through the same duct. Although considering SRLGs help to strengthen the network survivability, it makes the network protection problem much more difficult mainly because of the two sets of additional constraints it imposes, namely 1) a working lightpath and its backup lightpath cannot go through any common SRLG (whereas in the classic network protection problem, we only have to ensure that they do not go through any common link); and 2) if two working lightpaths go through a common SRLG, their backup lightpaths cannot share any network resource (while in the classic protection problem, similar constraints only apply to working paths going through the same link).

The SPP problem can be modeled as an optimization problem in a simple directed graph given the following.

- V: The set of nodes in the graph under consideration.
- E: The set of directed links (edges) in the graph.
- W: The number of wavelengths available on each link. The wavelengths are numbered from 1 to W.
- R: The set of connection requests. $M = |R|$. The requests are numbered from 1 to M. Each connection request has a source node and a destination node.[1] It requires a working lightpath and a backup lightpath from the stated source to the stated destination.
- G: The set of SRLGs. Each SRLG contains a set of links in E. Links in the same SRLG share the same risk, i.e., these links may break at the same time due to a destructive event. If two paths in the network have links in the same SRLG, we say that they are SRLG-joint. Otherwise, we call them SRLG-disjoint.

The goal is to determine a working lightpath and a backup lightpath for each connection request in R. The constraints are enumerated as follows.

C1: The number of wavelengths used on each link cannot not exceed W.

C2: The working lightpath and the backup lightpath, for each connection request, must be SRLG-disjoint.

C3: Two working lightpaths cannot use the same wavelength on the same link.

[1]Two different requests in C may have the same source and destination.

C4: A backup lightpath cannot share the same wavelength on the same link with any working lightpath.

C5: If two working lightpaths are SRLG-joint, their backup lightpaths cannot use the same wavelength on the same link.

The objective is to minimize the cost

$$\sum_{e \in E}(F_e + S_e) \tag{1}$$

where F_e is the number of wavelengths on link e used in working lightpaths, and S_e the number of wavelengths on link e used in backup lightpaths.

This problem has been studied and modeled as an integer linear programming (ILP) problem in [27]. It is an NP-complete problem [34]. An ILP approach involves too many constraints and variables, even for a small-sized network [35]. Therefore, dealing with any practical-sized network needs to resort to heuristics. Since the objective in this problem is to minimize the total number of wavelengths used in a routing scheme, a heuristic for it should have the following properties.

- The wavelength sharing among the backup lightpaths should be maximized.
- SRLG-disjointness among the working lightpaths should be encouraged so that backup lightpaths have a good chance of sharing wavelengths.
- The lightpaths should be as short as possible since a wavelength needs to be assigned to each link traversed by these lightpaths.

Compared to the extensive researches on dynamic protection where connection requests arrive one by one and future arrivals are not known (e.g., [36] and [37]), existing results on highly difficult offline SPPs, which handle all the connection requests to achieve global optimization objectives, are very limited. Specifically, [29] and [38] proposed to restrict the searching space of the ILP problem to a set of predefined alternative routes between each pair of nodes. The study in [39] applied the Lagrangean relaxation method to tackle the ILP formulations. Several network flow-based heuristics have also been proposed (e.g., [40]), where a good summary can be found in [41]. Other related yet different problems include the virtual topology design, where the wavelength of a connection can be changed on a limited set of intermediate nodes [42], [43], the length-limited hybrid protection problem [41], etc.

A heuristic for the SPP problem defined in this section has been proposed recently in [27]. To the best of our knowledge, it is the only practical heuristic for the problem in the literature. It first computes a working path and a backup path for each connection request. Then, it assigns a wavelength to each path. Finally, it rearranges the lightpaths to minimize the total number of the wavelengths used. It is easy to implement and have very low computational overheads. Its solution quality, although claimed to be suboptimal, can still be significantly improved as we will see later. Moreover, the heuristic of Zang–Ou–Mukherjee has no control parameters; therefore, it cannot be used in an indirect encoding EA.

One purpose of our study is to develop an indirect encoding EA using construction heuristics for producing solutions of high quality for complex optimization problems such as the SPP problem at modest computational costs.

III. Basic Heuristic (BH)

The proposed heuristic consists of three phases. In the first phase, a heuristic called the "working path router" ("Wrouter") routes a working path w_r for each connection request r in R. In the second phase, a heuristic called the "protection path router" ("Prouter") computes backup paths for all the working lightpaths established in the first stage. In the third phase, a heuristic called the "wavelength assigner" ("Wassigner") assigns a wavelength to each lightpath generated in the first two phases.

A. Wrouter

Let $P = \{p_1, p_2, \ldots, p_k\}$ be a set of paths in the graph. The SRLG-disjoint degree of path p_i in P, i.e., degree (p_i, P), is defined as the number of paths in P that are SRLG-disjoint from p_i. The SRLG-disjoint degree of P is defined as $\max_{p_i \in P}$ degree (p_i, P). Wrouter aims at establishing a set of working lightpaths of as large SRLG-disjoint degree as possible.

For each link e in the graph, its SRLG factor is defined as the number of SRLGs containing e. To increase the SRLG-disjoint degree of the set of working lightpaths, Wrouter discourages its working lightpaths from using links of high SRLG factors. The reason is that a path with links of high SRLG factor is very likely to be SRLG-joint with other paths.

Since working lightpaths cannot share the same wavelength on the same link and the number of wavelengths used on each link cannot exceed W, a single link, if possible, is prevented from being used by more than W lightpaths in Wrouter.

Wrouter routes working paths one by one in order of the connection requests $\pi = (\pi_1, \pi_2, \ldots, \pi_M)$. Suppose that for connection requests $\pi_1, \ldots, \pi_{k-1} (1 \leq k < M)$, a set of working lightpaths $\mathrm{WP}^{k-1} = \{\mathrm{wp}_{\pi_1}, \ldots, \mathrm{wp}_{\pi_{k-1}}\}$ has been established (in the case $k = 0$, no working lightpath has been established). If Wrouter failed in routing a working lightpath for request $r < k$, wp_r is set to be the empty path $\varnothing$. Wrouter routes the working lightpath wp_{π_k} for the connection request π_k in the following way.

Step 1) For each link in E, count the number of working lightpaths $\in \mathrm{WP}^{k-1}$ using it. Let $E_1 \subset E$ contain all the links that have been used by W lightpaths in WP^{k-1}.

Step 2) Compute the SRLG-disjoint degree of each nonempty working lightpath in WP^{k-1}. Let E_2 contain the links in $E \setminus E_1$ that are
a) in the nonempty working lightpaths in WP^{k-1} with the smallest SRLG-disjoint degree;
b) SRLG-joint with links specified in a).

Step 3) For each link $e \in E \setminus (E_1 \cup E_2)$, set its length to be

$$L(e) = (1 + c)^{\beta_e} \tag{2}$$

where β_e is its SRLG factor, and $c > 0$ is a control parameter.

For any link f in E_2, set its length to be

$$L(f) = \sum_{e \in E \setminus (E_1 \cup E_2)} L(e). \tag{3}$$

Step 4) Applying the Dijkstra algorithm [44] to the graph $(V, E \setminus E_1)$ with the link lengths defined in (2) and (3), compute the shortest path from the source to the destination of the connection request π_k as the working lightpath wp_{π_k}. If the Dijkstra algorithm fails in finding a path, set $\mathrm{wp}_{\pi_k} = \varnothing$.

The control parameters $\pi = (\pi_1, \pi_2, \ldots, \pi_M)$ and c can be regarded as the input to Wrouter, and the working lightpaths $\mathrm{WP} = \{\mathrm{wp}_1, \mathrm{wp}_2, \ldots, \mathrm{wp}_M\}$ are its output; therefore, Wrouter can be denoted by

$$\mathrm{WP} = \mathrm{Wrouter}(\pi, c). \tag{4}$$

Remark 1: If the lightpath wp_{π_k} uses any link in E_1, it will lead to an infeasible routing. This is why it is forbidden to use such links in wp_{π_k}. If the lightpath wp_{π_k} uses any link in E_2, the SRLG-disjoint degree of the resultant set of working lightpaths may be decreased. For this reason, Wrouter discourages wp_{π_k} from using any link in E_2. In fact, due to the link length settings in (2) and (3), wp_{π_k} will not use any link in E_2 if there is a path from the source node to the destination node of the request π_k in graph $(V, E \setminus (E_1 \cup E_2))$.

Remark 2: In (2), the higher the SRLG factor of a link is, the longer it is. In this way, Wrouter encourages wp_{π_k} to use links with low SRLG factors. Therefore, the SRLG-disjoint degree of the resultant set of working paths can be lowered.

Remark 3: For some control parameter settings, if there is no path from the source node to the destination node of the request π_k in the graph $(V, E \setminus E_1)$, then wp_{π_k} will be set to $\varnothing$ in Wrouter.

B. Prouter

Given a set of working lightpaths $\mathrm{WP} = \{\mathrm{wp}_1, \mathrm{wp}_2, \ldots, \mathrm{wp}_M\}$ for all the connection requests in R (where wp_r is for request r, and some wp_i in WP may be empty), the goal of Prouter is to route the backup lightpath bp_r for each connection request r.

In a similar way to Wrouter, Prouter establishes backup lightpaths one by one in order of the connection requests $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_M)$. Suppose that for connection requests $\sigma_1, \ldots, \sigma_{k-1} (1 \le k < M)$, a set of backup lightpaths $\mathrm{BP}^{k-1} = \{\mathrm{bp}_{\sigma_1}, \ldots, \mathrm{bp}_{\sigma_{k-1}}\}$ has been established (in the case $k = 0$, no backup lightpath has been established). Then Prouter routes the backup lightpath bp_{σ_k} for the connection request σ_k in the following way.

Step 1) If $\mathrm{wp}_{\sigma_k} = \varnothing$, set $\mathrm{bp}_{\sigma_k} = \varnothing$. Otherwise, go to Step 2).

Step 2) Let Q be the set containing all the backup lightpaths in BP^{k-1} whose corresponding working lightpaths are SRLG-joint with the working lightpath wp_{σ_k}. Let H contain links that
- a) have been used by W lightpaths in $Q \cup \mathrm{WP}$;
- b) have been used by the working lightpath wp_{σ_k};
- c) are SRLG-joint with links in the working path wp_{σ_k}.

For any link $e \in E \setminus H$, set its length to be

$$L(e) = 1. \tag{5}$$

Applying the Dijkstra algorithm to the graph $(V, E \setminus H)$ with the link lengths defined in (5), compute the shortest path from the source to the destination of the connection request σ_k as the backup lightpath bp_{σ_k}. If the Dijkstra algorithm fails in finding a path, set $\mathrm{bp}_{\sigma_k} = \varnothing$.

The control parameters $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_M)$ and the set of working lightpaths WP can be regarded as the input to Prouter, and the backup lightpaths $\mathrm{BP} = \{\mathrm{bp}_1, \mathrm{bp}_2, \ldots, \mathrm{bp}_M\}$ are its output. Therefore, Prouter can be denoted by

$$\mathrm{BP} = \mathrm{Prouter}(\sigma, \mathrm{WP}). \tag{6}$$

Remark 1: If the lightpath bp_{σ_k} uses any links in H defined in Step 3), it will lead to an infeasible routing. For this reason, Prouter prevents bp_{π_k} from using such links.

Remark 2: If there is no path from the source node to the destination node of the request σ_k in the graph $(V, E \setminus H)$ or wp_{σ_k}, Prouter will set $\mathrm{bp}_{\sigma_k} = \varnothing$.

C. Wassigner

Given the set of working lightpaths WP and the set of backup lightpaths BP for all the connection requests in R, the task of Wassigner is to assign wavelengths to every working lightpath and backup lightpath such that the total number of wavelengths used is minimized under the constraints listed as follows.

CW1: Two working lightpaths must be assigned different wavelengths if they traverse the same link.

CW2: A working path and a backup lightpath must be assigned different wavelengths if they traverse the same link.

CW3: Two backup lightpaths have to be assigned different wavelengths if they traverse the same link and their corresponding working lightpaths are SRLG-joint.

Viewing each lightpath as a node in a graph and each wavelength as a color, two nodes (lightpaths) are defined to be adjacent if they must be assigned different wavelengths. Then, the above problem becomes the well-known NP-hard graph coloring problem [45]. We use a first-fit heuristic for this problem. We assume that the number of wavelengths available are infinite and that the wavelengths are indexed by 1, 2, ... The heuristic, Wassigner, works as follows.

Step 1) Set $P = \mathrm{BP} \cup \mathrm{WP}$. Remove all the empty paths from P.

Step 2)
- Step 2.1) Remove a lightpath p from P.
- Step 2.2) Assign the allowable wavelength with the smallest index to p. A wavelength is allowable if assigning it to p does not violate CW1–CW3.

Step 3) If $P = \emptyset$, stop. Otherwise, go to Step 2).

There are several ways in Step 2.1) to select from P a lightpath to remove. In our implementation, we randomly pick a lightpath from P and remove it in Step 2.1). WP and BP are inputs to Wassigner. While the output is a wavelength assignment to each lightpath in $P = \text{BP} \cup \text{WP}$, $A : P \rightarrow \{1, 2, \ldots\}$. Wassigner can be denoted by

$$A = \text{Wassigner}(\text{WP}, \text{BP}). \tag{7}$$

Remark 1: The wavelength assignment A generated in Wassigner will satisfy the constraints CW1–CW3. However, if the number of wavelengths used in A exceeds W, it may violate the constraint C1 (i.e., the number of wavelengths used on some paths may exceed W).

D. Structure of Heuristic Algorithm

For the SPP problem, the proposed BH has the control parameters $\pi = (\pi_1, \pi_2, \ldots, \pi_M)$, $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_M)$, and c. The output is a working lightpath set WP, a backup lightpath set BP, and a wavelength assignment $A : P \rightarrow \{1, 2, \ldots\}$. It works as

$$(\text{BP}, \text{WP}, A) = \text{BH}(\pi, \sigma, c)$$

where $\text{WP} = \text{Wrouter}(\pi, c)$, $\text{BP} = \text{Prouter}(\sigma, \text{WP})$, and $A = \text{Wassigner}(\text{WP}, \text{BP})$.

Any solution generated by BH will satisfy constraints C2–C5. An inappropriate setting of π, σ, and c may generate a solution that is incomplete (there are no working lightpaths or backup lightpaths for some requests) and/or violate constraint C1. It is not easy to develop a simple mathematical model for the relationship between a setting of algorithm parameters π, σ, and c and the quality of the solution constructed. Finding an optimal parameter setting turns out to be a black box optimization problem. An EA/G and a GA are used as tools for tuning the control parameters π and σ in our study.

IV. EA/G

Offspring generators play a crucial role in any EA. The proximate optimality principle [46], an underlying assumption in most (if not all) heuristics, assumes that good solutions have similar structure. This assumption is reasonable for most real-world problems. Based on this assumption, an ideal offspring assumption should be able to produce a solution that is close to the best solutions found so far. Only a few parent solutions are involved in crossover and mutation. A new solution generated by these two operators may be far from other best solutions found so far in the search. However, in EDAs [21], new solutions are sampled from a model that characterizes a promising area in the search space, but there is no mechanism in EDAs for directly controlling the similarity (often measured by distance) between new solutions and the best solutions found so far. Guided mutation combines global statistical information and location information of the best solutions found so far to overcome the shortcomings of GAs and EDAs. In the following, we present EA/G, which is used in our study for tuning the control parameters in the BH defined in the previous section.

A. Search Space and Objective Function

The search space in EA/G is Π, the set of all possible permutations of $I = \{1, 2, \ldots, M\}$. Each permutation has a cost. The task of EA/G is to find a permutation solution in Π with the (nearly) lowest cost.

B. Population and Probability Matrix

At each generation t, EA/G maintains a population of N solutions (i.e., permutations of I)

$$\text{Pop}(t) = \{\xi^1, \xi^2, \ldots, \xi^N\}$$

and a probability matrix

$$X(t) = \begin{pmatrix} x_{11}(t) & \cdots & x_{1M}(t) \\ \vdots & \ddots & \vdots \\ x_{M1}(t) & \cdots & x_{MM}(t) \end{pmatrix}$$

where $X(t)$ is the distribution of promising solutions in the search space. More precisely, $x_{ij}(t)$ is the probability that $\xi_i = j$ in a promising permutation solution $\xi = (\xi_1, \xi_2, \ldots, \xi_M)$.

C. Initialization

EA/G randomly chooses N permutations from Π as its initial population $P(0)$. The initial probability matrix $X(0) \in R^{M \times M}$ is set as

$$X(0) = \begin{pmatrix} \frac{1}{M} & \cdots & \frac{1}{M} \\ \vdots & \ddots & \vdots \\ \frac{1}{M} & \cdots & \frac{1}{M} \end{pmatrix}. \tag{8}$$

D. Update of Probability Matrix

Assume that the population at generation t is $\text{Pop}(t) = \{\xi^1, \xi^2, \ldots, \xi^N\}$, and the probability matrix at generation $t-1$ is $X(t-1)$. Then, the probability matrix $X(t) = (x_{ij}(t))_{M \times M}$ can be computed as

$$x_{ij}(t) = (1-\beta)\frac{1}{N}\sum_{k=1}^{N} I_{ij}(\xi^k) + \beta x_{ij}(t-1), \qquad (1 \le i, j \le M) \tag{9}$$

where

$$I_{ij}(\xi) = \begin{cases} 1, & \text{if } \xi(i) = j \\ 0, & \text{otherwise.} \end{cases}$$

$0 \le \beta \le 1$ is the learning rate. The bigger β is, the greater is the contribution of the solutions in $\text{Pop}(t)$ to the probability matrix $X(t)$.

E. Generation of New Solutions: Guided Mutation

Guided by the probability matrix $X = (x_{ij})_{M \times M}$, the guided mutation [25], [26] mutates an existing solution ξ to generate a new solution ζ. This operator also needs a control parameter $0 < \alpha < 1$. It works as follows:

$$\zeta = \text{Guided Mutation } (\xi, X, \alpha).$$

Input: A permutation $\xi = (\xi_1, \dots, \xi_M)$, a probability matrix $X = (x_{ij})_{M \times M}$, and a positive parameter $\alpha < 1$.
Output: $\zeta = (\zeta_1, \dots, \zeta_M)$, a permutation.

Step 1) Randomly pick $[\alpha M]$ integers from $I = \{1, 2, \dots, M\}$, and let these integers constitute a set $K \subset I$. Set $V = I \setminus K$ and $U = I \setminus \{\xi_l | l \in K\}$.
Step 2) For each $i \in K$, set $\zeta_i = \xi_i$.
Step 3) While $(U \neq \emptyset)$, uniformly randomly select i from V, and then randomly draw k from U with probability

$$\frac{x_{ik}}{\sum_{j \in U} x_{ij}}.$$

Set $\zeta_i = k$, $U = U \setminus \{k\}$ and $V = V \setminus \{i\}$.
Step 4) Return ζ.

In the above guided mutation operator, ζ_i is directly copied from the parent ξ if $i \in K$. Otherwise, under the constraint that ζ is a permutation, it is randomly generated based on the probability matrix X. The larger α is, the more elements of ζ are directly copied from its parent ξ. In other words, α controls the similarity between the offspring and the parent.

In the conventional mutation for permutation vectors, several (often two) elements are randomly selected and then swapped. The probability that a permutation ζ being generated from parent ξ is entirely determined by the number of the positions where ζ and ξ are different. In contrast, the guided mutation mutates ξ based on the probability matrix X, which is learned and updated at each generation for modeling the distribution of promising solutions. It can be expected that offspring ζ fall in or close to a promising area in the search space. Meanwhile, randomness in Step 3) also provides diversity for the search.

F. Structure of EA/G

EA/G works as follows.

Step 0) Parameter Setting: Set the following control parameters:
- $PopSize$: population size;
- $NoNew$: number of new solutions generated at each generation;
- α: control parameter in Guided Mutation;
- β: learning rate used in the update of the probability matrix;
- $MaxGen$: maximum number of generations EA/G runs for.

Step 1) Initialization: Set $t := 0$. Initialize the probability matrix $X(t)$ by (8). Randomly generate $Popsize$ distinct permutations to form the initial population Pop(0). Let the best solution in Pop(0) be ξ^*.
Step 2) Guided Mutation: Independently apply the guided mutation operator to ξ^* (i.e., $\xi^*, X(t)$, and α are inputs to the guided mutation) $NoNew$ times to generate $NoNew$ solutions.
Step 3) Selection: Select the $PopSize$ distinct best solutions from the solutions generated in Step 2) and Pop(t). Let these selected solutions form Pop$(t+1)$. Let the best solution in Pop$(t+1)$ be ξ^*.
Step 4) Stopping Condition: If $t = MaxGen$, stop and output the best solution ξ^*; otherwise, set $t = t+1$.
Step 5) Update of Probability Matrix: Compute $X(t)$ by (9) and then go to Step 2).

In this algorithm, the members of the population Pop(t) are always distinct. This prevents $X(t)$ from becoming degenerate (i.e., all the elements in it are very close to 0 or 1) and thus provides diversity for the search. In Step 2), the guided mutation operator is always applied to the best solution ζ^*. Therefore, the new solutions generated are, to some extent, similar to the best solution found so far, which is desirable for a successful heuristic according to the proximate optimality principle [46]. In Step 3), the quality of a solution is measured by its cost. The smaller its cost is, the better it is.

V. GA

This section describes the GA used in our study for tuning the control parameters in BH. The search space is the permutation vector space.

Step 0) Parameter Setting: Set the following control parameters:
- $PopSize$: population size;
- $NoNew$: number of new solutions generated at each generation;
- μ and γ: control parameters in mutation operator;
- $MaxGen$: maximum number of generations the algorithm runs for.

Step 1) Initialization: Set $t := 0$. Randomly generate $Popsize$ distinct permutations to form the initial population Pop(0). Let the best solution in Pop(0) be ξ^*.
Step 2) Crossover and Mutation: Randomly select $NoNew$ pairs of permutations from Pop(t) and perform the cycle crossover (CX) operator [8] on each pair to generate $NoNew$ new permutations. For each solution generated in Step 2), generate a uniform random number $rand$ in $(0, 1)$. If $rand < \mu$, mutate it by performing $[\gamma M]$ random swaps on it.
Step 3) Selection: Select the $PopSize$ distinct best solutions (i.e., with the smallest costs) from the solutions generated in Step 2) and Pop(t). Let these selected solutions form Pop$(t+1)$.
Step 4) Stopping Condition: If $t = MaxGen$, stop and output the best solution ξ^* in Pop$(t+1)$. Otherwise, set $t = t+1$, and go to Step 2).

Except for the mechanisms for generating new solutions and initialization, all the other components in the above GA are the same as in EA/G. The reason that we chose the CX operator is that this operator works well for other combination optimization problems [8].

TABLE I
PARAMETERS OF THE TEST NETWORKS

	No. of Nodes	No. of Links	No. of SRLGs	No. of Requests	No. of Wavelengths Available
Setting 1	19	62	31	100	32
Setting 2	19	62	29	100	32
Setting 3	19	62	27	100	32
Setting 4	24	86	43	100	32
Setting 5	24	86	41	100	32
Setting 6	24	86	40	100	32
Setting 7	24	86	39	100	32
Setting 8	31	94	47	150	64
Setting 9	50	200	85	250	64
Setting 10	50	200	85	275	64
Setting 11	50	200	85	300	64
Setting 12	60	200	90	250	64
Setting 13	65	200	100	250	64
Setting 14	65	200	100	300	64
Setting 15	100	340	150	350	64

VI. Tuning the Parameter Setting of Heuristic: BH/EA/G and BH/GA

The proposed heuristic BH may generate an incomplete solution or a solution violating constraint C1. We need to measure its performance in tuning the control parameters in BH. For a solution $(\mathrm{BP}, \mathrm{WP}, A)$ generated by BH, we use the following function as its cost function:

$$2M|E|(N_1 + N_2) + \sum_{e \in E}(F_e + S_e) \qquad (10)$$

where F_e is the number of wavelengths on link e used in working lightpaths, and S_e the number of wavelengths on link e used in backup lightpaths. M is the number of the connection requests, $|E|$ is the number of all the links in E, N_1 is the number of empty lightpaths in BP $\cup$ WP, and

$$N_2 = \begin{cases} 0, & \text{if } W \geq K \\ K - W, & \text{otherwise} \end{cases}$$

where K is the number of wavelengths used in A.

Obviously, due to the penalty term in (10), the cost of an infeasible or incomplete solution is always higher than that of a feasible solution.

For each parameter setting of (π, σ, c) in BH, we define its cost $\mathrm{cost}(\pi, \sigma, c)$ to be the cost of the solution generated by BH with such a parameter setting. The proposed procedure for tuning these parameters works as follows.

BH/EA/G (BH/GA)

Step 1) Tuning c.

Step 1.1) Randomly generate ten pairs of permutations $(\pi^1, \sigma^1), (\pi^2, \sigma^2), \ldots, (\pi^{10}, \sigma^{10})$. Let $c_i = (i/10)(i = 1, 2, \ldots, 10)$.

Step 1.2) Compute

$$c^\star = \arg \min_{c \in \{c_1, c_2, \ldots, c_{10}\}} \frac{1}{10} \sum_{j=1}^{10} \mathrm{cost}(\pi^j, \sigma^j, c).$$

Step 2) Tuning π.

Step 2.1) Randomly generate a permutation $\tilde{\sigma}$.

Step 2.2) Use EA/G (GA) to tune π, where the cost of π is set as $\mathrm{cost}(\pi, \tilde{\sigma}, c^\star)$. Set $\pi^\star$ to be the best setting of π found in EA/G (GA).

Step 3) Tuning σ.
Use EA/G (GA) to tune σ, where the cost of σ is set as $\mathrm{cost}(\pi^\star, \sigma, c^\star)$. Set $\sigma^\star$ to be the best setting of σ found in EA/G (GA).

Then, the best solution found is the one generated by BH with parameter setting $(\pi^\star, \sigma^\star, c^\star)$.

We can iterate the above procedure many times in order to lower the cost of the solution obtained as in the alternating variable optimization method [47]. Taking the computational overhead into consideration, we chose not to perform the iteration in our experimental study.

VII. Experimental Results

A. Test Networks

The basic characteristics of the test network instances are listed in Table I.

For each network parameter setting, we test two network examples. In the first one, the links in the same SRLG are adjacent. In the second one, the links in each SRLG are randomly selected from E. The first test example with parameter setting a is denoted by $T1 - a$, and the second test example is denoted by $T2 - a$.

A test network example is generated as follows.

1) Let $G = (V, E)$. E has N_L distinct links that are randomly selected.
2) Randomly divide all the links in E into $|G|$ groups. Each group is an SRLG. In the case of test example $T1 - a$, the links in the same SRLG should be adjacent. In other words, let H be the subgraph induced by SRLG, then H should be connected if the directions of the links are not considered.
3) Randomly select M connection requests. Let the number of wavelengths available on each link be W.
4) Apply the heuristic of Zang–Ou–Mukherjee [27] to the generated problem. If a feasible solution can be found, then stop. This problem will be taken as a test network instance. Otherwise, go to Step 1).

The above procedure can produce an instance of the SPP problem that is guaranteed to have at least one feasible solution. As pointed out in [27], linear programming methods are not suitable for dealing with networks as large as the above networks.

B. Parameter Setting in EA/G and GA in Comparison

In our experimental study, we set $NoNew = 100$ and $MaxGen = 100$ in both EA/G and GA. Therefore, both BH/EA/G and BH/GA need to call BH 20 000 times [10 000 times for tuning π in Step 2) of BH/EA/G (BH/GA) and 10 000 times for tuning σ].

We set $Popsize = 50$ in EA/G.

In EA/G, there are another two parameters, i.e., α (in the guided mutation operator) and β (used in updating the probability matrix), while GA has three more parameters to set, i.e., μ, γ used in the mutation operator, and $PopSize$ used for controlling the selection pressure. To determine an appropriate setting for these parameters, we have run BH/EA/G and BH/GA on $T1-1$ for $\alpha, \beta = 0.0, 0.1, \ldots, 0.9$, and $\mu, \gamma = 0.0, 0.1, \ldots, 0.9$, $Popsize = 20, 30, 40, 50, 60, 70, 80$, respectively. We have found that $\alpha = 0.1$ and $\beta = 0.2$ is the best setting for EA/G and $\mu = 0.1$, $\gamma = 0.3$, and $Popsize = 40$ is the best for GA for $T1-1$. In the following comparison study, we always use these settings for EA/G and GA, although it does not mean that these settings are the best for all the test problems.

C. Comparison Results

We have compared the following four algorithms in our experimental study:

1) BH/EA/G;
2) BH/GA;
3) BH/R, which has the same structure as BH/EA/G and BH/GA, except that it tests 10 000 random permutations for π in Step 2) and 10 000 random permutations for σ in Step 3), respectively;
4) ZOM-H, heuristic of Zang–Ou–Mukherjee.

The first three algorithms need to call BH the same number of times in each run. Due to the computational cost, each BH/EA/G, BH/GA, and BH/R has been run independently for ten times on each test network instance.

Tables II and III shows the experimental results including the following:

- *time*: The average run time (in seconds) of each algorithm for each test instance.
- *cost*: The lowest cost found in the heuristic of Zang–Ou–Mukherjee. Since there is no randomness in this heuristic, we only run it once for obtaining its $cost$.
- *best*: The lowest cost found in ten independent runs for each test instance by BH/EA/G, BH/GA, and BH/R, respectively.
- *avg*: The average of the solution costs in ten independent runs for BH/EA/G, BH/GA, and BH/R, respectively.
- *std*: The standard derivation of the lowest costs found in ten independent runs for BH/EA/G, BH/GA, and BH/R, respectively.

All the experiments were performed on a cluster of Athlon MP 1900s (1.6 GHz). It is from Table II that the running time

TABLE II
AVERAGE RUN TIME (IN SECONDS) OF HA, BH/R, BH/GA, AND BH/EA/G ON TEST NETWORK INSTANCES

instances	ZOM-H	BH/R	BH/GA	BH/EA/G
T_1-1	2.51	76.40	101.80	141.80
T_1-2	2.74	77.10	99.50	158.50
T_1-3	3.10	80.60	139.50	161.30
T_1-4	2.13	82.80	129.60	149.10
T_1-5	2.01	77.50	114.40	136.70
T_1-6	3.10	77.60	141.20	159.60
T_1-7	2.32	70.40	135.30	127.60
T_1-8	41.10	675.10	874.10	1180.30
T_1-9	63.30	1586.30	2005.00	2601.70
T_1-10	150.60	2115.70	3133.00	3492.50
T_1-11	161.60	2010.90	3512.70	3954.20
T_1-12	165.40	2174.60	3599.10	3741.25
T_1-13	200.10	2875.10	4112.80	4385.90
T_1-14	241.10	8081.90	12091.53	14951.50
T_1-15	246.60	7651.40	12109.51	13388.40
T_2-1	2.55	82.10	127.50	151.30
T_2-2	2.69	89.60	147.20	161.40
T_2-3	2.05	71.30	128.70	139.60
T_2-4	2.96	97.80	168.10	187.40
T_2-5	2.29	85.10	120.60	151.40
T_2-6	2.67	79.80	154.50	179.50
T_2-7	2.81	75.10	142.50	168.70
T_2-8	46.40	689.50	671.70	848.40
T_2-9	67.90	1480.60	2000.10	1861.50
T_2-10	152.80	2061.50	2975.80	3364.90
T_2-11	164.20	2513.60	3552.90	3998.90
T_2-12	163.50	2315.50	3132.50	3776.20
T_2-13	210.60	2910.60	4010.90	4408.80
T_2-14	251.70	7916.90	11257.90	14225.40
T_2-15	239.80	7412.30	11219.40	13395.90

for the heuristic of Zang–Ou–Mukherjee is from a few seconds to about 4 min for the test network instances, while the other three algorithms need from 2 min to 4 h for each run. For offline applications that typically plan the allocations of connection requests for a long period of time (days or even months), a 4-h computation time is totally acceptable.

Table III clearly shows that BH/EA/G, BH/GA, and BH/R are much better than the heuristic of Zang–Ou–Mukherjee in terms of solution quality.

Table III also shows that *std* are the about the same for BH/EA/G, BH/GA, and BH/R for the test instances, and BH/EA/G and BH/GA perform significantly better than BH/R for all the test problems with the same number of BH calls in terms of *best* and *avg*. These results imply that EA's search mechanism is very effective in tuning the control parameters in BH. Perhaps this is because the proximate optimality principle also holds for the BH control parameters as in many other real-world problems.

It is evident from Table III that BH/EA/G performs better than BH/GA in all the test instances but T_1-7. These results suggest that the combination of global statistical information and location information in the guided mutation operator does improve the performance of the search.

Figs. 1 and 2 give the evolution of the average cost of the best solutions found in ten runs with the number of BH calls in tuning π and σ on two test instances. The first 10 000 BH calls are for tuning π, while the last 10 000 calls are for tuning σ. It is very clear from these four figures that there are significant

TABLE III
SOLUTION QUALITY OF HA, BH/R, BH/GA, AND BH/EA/G ON TEST NETWORK INSTANCES

	ZOM-H	BH/R			BH/GA			BH/EA/G		
instances	cost	*best*	*avg*	*std.*	*best*	*avg*	*std.*	*best*	*avg*	*std.*
T_1-1	478.00	404.00	409.60	4.81	369.00	381.50	4.18	365.00	372.50	4.03
T_1-2	458.00	401.00	403.80	3.06	378.00	382.70	5.06	368.00	374.55	4.57
T_1-3	487.00	405.00	410.90	5.60	370.00	381.40	7.38	364.00	368.70	6.01
T_1-4	445.00	411.00	420.05	6.94	394.00	401.50	3.05	377.00	384.60	5.58
T_1-5	492.00	427.00	430.90	5.23	393.00	410.30	4.56	384.00	390.75	5.05
T_1-6	462.00	417.00	420.20	3.38	390.00	399.00	7.49	377.00	384.90	5.72
T_1-7	451.00	420.00	423.30	3.83	383.00	386.40	2.47	384.00	403.10	6.18
T_1-8	785.00	701.00	710.25	7.78	660.00	671.40	8.07	636.00	649.20	4.86
T_1-9	1389.00	1308.00	1314.15	5.41	1257.00	1268.40	8.10	1221.00	1239.10	0.00
T_1-10	1516.00	1467.00	1474.30	8.35	1407.00	1432.20	10.00	1348.00	1372.50	0.00
T_1-11	1612.00	1514.00	1519.20	7.66	1453.00	1467.50	7.64	1426.00	1439.20	0.00
T_1-12	1661.00	1494.00	1502.40	9.46	1447.00	1461.10	7.99	1414.00	1424.80	0.00
T_1-13	1834.00	1621.00	1636.40	5.19	1557.00	1581.90	10.76	1526.00	1551.60	0.00
T_1-14	2169.00	1924.00	1931.20	8.34	1829.00	1857.10	11.23	1809.00	1825.90	0.00
T_1-15	2511.00	2298.00	2328.70	10.57	2208.00	2253.70	12.57	2195.00	2215.00	0.00
T_2-1	452.00	392.00	398.75	5.41	367.00	371.70	3.02	355.00	364.70	3.33
T_2-2	458.00	383.00	394.25	10.50	364.00	376.50	3.77	355.00	363.45	3.53
T_2-3	501.00	436.00	440.05	11.10	410.00	419.60	6.50	399.00	404.50	5.13
T_2-4	479.00	433.00	440.10	8.60	410.00	417.40	6.51	399.00	406.30	4.15
T_2-5	444.00	405.00	408.60	7.50	375.00	390.50	3.24	366.00	376.35	6.13
T_2-6	467.00	422.00	426.65	9.00	396.00	405.90	4.76	380.00	388.35	3.26
T_2-7	540.00	462.00	467.25	5.10	435.00	444.90	5.93	419.00	428.90	4.85
T_2-8	898.00	762.00	776.75	10.50	736.00	741.50	5.01	715.00	727.80	6.72
T_2-9	1369.00	1264.00	1277.35	15.60	1223.00	1239.10	8.41	1192.00	1203.35	8.63
T_2-10	1415.00	1376.00	1383.10	15.80	1330.00	1353.20	7.34	1213.00	1225.70	7.57
T_2-11	1569.00	1518.00	1527.60	9.63	1475.00	1500.90	11.37	1397.00	1401.80	9.10
T_2-12	1649.00	1554.00	1560.00	7.89	1491.00	1501.40	12.53	1451.00	1476.90	11.41
T_2-13	1785.00	1654.00	1666.00	10.70	1609.00	1615.50	6.76	1567.00	1582.70	8.75
T_2-14	2114.00	1970.00	1984.40	15.10	1904.00	1917.70	13.60	1866.00	1889.70	10.50
T_2-15	2446.00	2265.00	2282.70	11.30	2198.00	2221.90	14.90	2155.00	2183.30	9.81

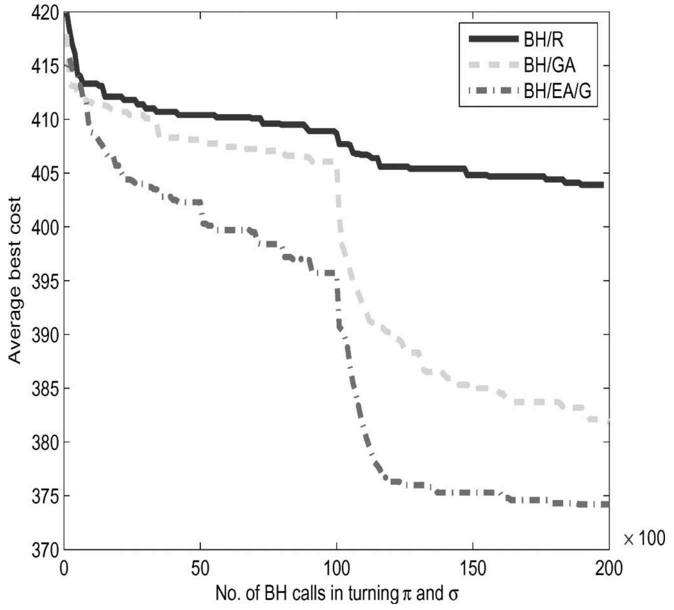

Fig. 1. Evolution of the average of the lowest solution costs with the number of BH calls in tuning π and σ in BH/R, BH/GA, and BH/EA/G for $T_1 - 2$.

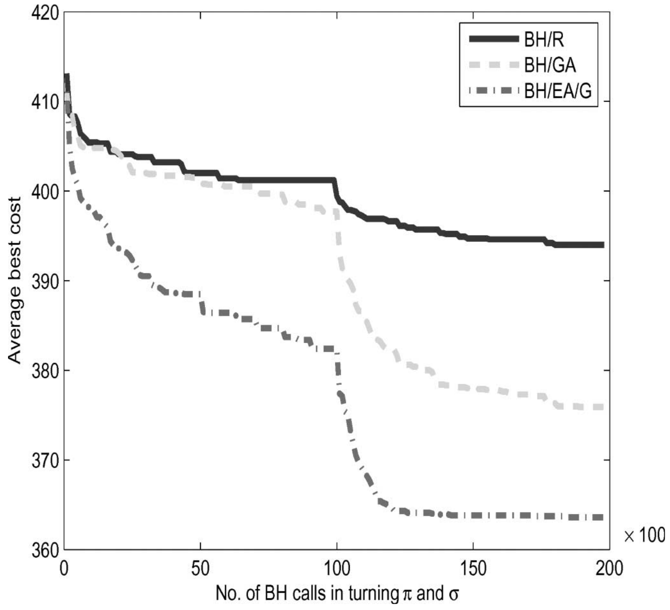

Fig. 2. Evolution of the average of the lowest solution costs with the number of BH calls in tuning π and σ in BH/R, BH/GA, and BH/EA/G for $T_2 - 2$.

decreases in the solution costs in both phases of tuning π and σ in BH/EA/G and BH/GA.

The test problems and the C++ code of all the algorithms in this paper can be found in cswww.essex.ac.uk/staff/zhang/.

VIII. CONCLUSION

Some real-world optimization problems may have complex data structures. Indirect encoding EAs using construction heuristics represent a feasible approach of dealing with such problems. In this paper, such an approach has been applied for solving the SPP problem in WDM networks with SRLGs. The proposed method optimizes the control parameters of a basic construction heuristic for the problem by using EA/G. The construction heuristic BH has three phases with three different parameters, which largely determine the performance of BH. Two parameters are permutations, and the third one is real valued. These three parameters are tuned separately to search for a nearly optimal solution to the problem. Our experimental study shows that the approach performs much better than the heuristic of Zang–Ou–Mukherjee. More significantly, by comparing the performances of EA/G, GA, and a pure random method for tuning the control parameters, we have shown that EAs are suitable for such a task, and the guided mutation operator does improve the performance of EAs.

In the future, we intend to apply this approach to solve other hard search and optimization problems.

Acknowledgment

The authors would like to thank for H. Zang for helpful discussions on the heuristic of Zang–Ou–Mukherjee and the SPP problem and J. Ford and G. Boshnakov for constructive comments.

References

[1] D. Corne, M. Dorigo, and F. Glover, Eds., *New Ideas in Optimisation*, New York: McGraw-Hill, 1999.

[2] H. E. William, E. N. Krasnogor, and J. E. Smith, Eds., *Recent Advances in Memetic Algorithms*, New York: Springer-Verlag, 2005.

[3] P. Moscato and C. Cotta, "A gentle introduction to memetic algorithms," in *Handbook of Meta-Heuristics*, F. Glover and G. Kochenberger, Eds. Norwell, MA: Kluwer, 2003, pp. 105–144.

[4] N. Krasnogor, "Studies on the theory and design space of memetic algorithms," Ph.D. dissertation, Univ. West England, Bristol, U.K., 2002.

[5] Y. S. Ong and A. J. Keane, "Meta-Lamarckian in memetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.

[6] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 392–403, Aug. 1998.

[7] Q. Zhang and Y.-W. Leung, "Orthogonal genetic algorithm for multimedia multicast routing," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 53–62, Apr. 1999.

[8] P. Merz and B. Freisleben, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Trans. Evol. Comput.*, vol. 4, no. 4, pp. 337–352, Nov. 2000.

[9] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 141–152, Feb. 2006.

[10] H.-L. Fang, P. M. Ross, and D. Corne, "A promising hybrid GA/heuristic approach for open-shop scheduling problems," in *Proc. 11th ECAI*, A. Cohn, Ed., 1994, pp. 590–594.

[11] E. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of Meta-Heuristics*, F. Glover and G. Kochenberger, Eds. Norwell, MA: Kluwer, 2003, pp. 457–474.

[12] N. Krasnogor and J. E. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy and design issues," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 474–488, Oct. 2005.

[13] B. A. Huberman, R. M. Lukose, and T. Hogg, "An economics approach to hard computational problems," *Science*, vol. 275, no. 5296, pp. 51–54, Jan. 3, 1997.

[14] A. Kapsalis, V. J. Rayward-Smith, and G. D. Smith, "Solving the graphical Steiner tree problem using genetic algorithms," *J. Oper. Res. Soc.*, vol. 44, no. 4, pp. 397–406, 1993.

[15] Y. Xiong, B. Golden, and E. Wasil, "A one-parameter genetic algorithm for the minimum labeling spanning tree problem," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 55–60, Feb. 2005.

[16] D. A. Pelta and N. Krasnogor, "Multimeme algorithms using fuzzy logic based memes for protein structure prediction," in *Recent Advances in Memetic Algorithms*. New York: Springer-Verlag, 2005, pp. 49–64.

[17] P. E. Hotz, "Comparing direct and developmental encoding schemes in artificial evolution: A case study in evolving lens shapes," in *Proc. IEEE Congr. Evol.*, 2004, pp. 752–757.

[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[19] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: Univ. of Michigan Press, 1975.

[20] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1996.

[21] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA: Kluwer, 2002.

[22] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-94-163, 1994.

[23] Q. Zhang and H. Muehlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 127–136, Apr. 2004.

[24] Q. Zhang, "On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 80–93, Feb. 2004.

[25] Q. Zhang, J. Sun, and E. P. K. Tsang, "Evolutionary algorithm with the guided mutation for the maximum clique problem," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 192–200, Apr. 2005.

[26] Q. Zhang, J. Sun, E. P. K. Tsang, and J. A. Ford, "Combination of guided local search and estimation of distribution algorithm for solving quadratic assignment problem," in *Proc. Bird Feather Workshops, Genetic and Evol. Comput. Conf.*, 2004, pp. 42–48.

[27] H. Zang, C. Ou, and B. Mukherjee, "Path-backup routing and wavelength-assignment (RWA) in WDM mesh networks under duct-layer constraints," *IEEE/ACM Trans. Netw.*, vol. 11, no. 2, pp. 248–258, Apr. 2003.

[28] B. Mukherjee, *Optical Communication Networks*. New York: McGraw-Hill, 1997.

[29] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh network, Part I—Protection," in *Proc. IEEE Infocom*, 1999, pp. 744–751.

[30] C. Ou, J. Zhang, H. Zang, L. H. Sahasrabuddhe, and B. Mukherjee, "New and improved approaches for shared-path protection in WDM mesh networks," *J. Lightw. Technol.*, vol. 22, no. 5, pp. 1223–1232, May 2004.

[31] Y. Xiong, D. Xu, and C. Qiao, "Achieving fast and bandwidth efficient shared-path protection," *J. Lightw. Technol.*, vol. 21, no. 2, pp. 365–371, Feb. 2003.

[32] J. Strand, A. Chiu, and R. Tkach, "Issues for routing in the optical layer," *IEEE Commun. Mag.*, vol. 39, no. 2, pp. 81–87, Feb. 2001.

[33] P. Sebos, J. Yates, A. Greenberg, and D. Rubenstein, "Effectiveness of shared risk link group auto-discovery in optical networks," in *Proc. Opt. Fiber Commun. Conf.*, Mar. 2002, pp. 493–495.

[34] J. Q. Hu, "Diverse routing in optical mesh networks," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 489–494, Mar. 2003.

[35] H. Zang, J. P. Jue, and B. Mukherjeee, "A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks," *Opt. Netw. Mag.*, vol. 1, no. 1, pp. 47–60, Jan. 2000.

[36] S. Sengupta and R. Ramamurthy, "From network design to dynamic provisioning and restoration in optical cross-connect mesh networks: An architectural and algorithmic overview," *IEEE Netw.*, vol. 15, no. 4, pp. 46–54, Jul./Aug. 2001.

[37] G. Ellinas, E. Bouillet, R. Ramamurthy, J. Labourdette, S. Chauduri, and K. Bala, "Routing an restoration architectures in mesh optical networks," *SPIE Opt. Netw. Mag.*, vol. 4, no. 1, pp. 91–106, Jan./Feb. 2003.

[38] M. Sridharan, A. Somani, and M. Salapaka, "Approaches for capacity and revenue optimization in survivable WDM networks," *J. High Speed Netw.*, vol. 10, no. 2, pp. 109–124, 2001.

[39] D. Doshi *et al.*, "Optical network design and restoration," *Bell Labs Tech. J.*, vol. 4, no. 1, pp. 58–84, 1999.

[40] P. Ho and H. Mouftah, "Reconfigurations of spare capacity for MPLS-based recovery for the Internet backbone networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 73–84, Feb. 2004.

[41] L. Shen, X. Yang, and B. Ramamurthy, "Shared risk link group (SRLG)-diverse path provisioning under hybrid service level agreement in wavelength-routed optical mesh networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 4, pp. 918–931, Aug. 2005.

[42] P. Datta, M. T. Frederick, and A. K. Somani, "Sub-graph routing: A novel fault-tolerant architecture for shared-risk link group failures in WDM optical networks," in *Proc. DRCN*, 2003, pp. 296–303.

[43] A. Todimala and B. Ramamurthy, "Survivable virtual topology routing under shared risk link groups in WDM networks," in *Proc. BoardNets*, 2004, pp. 130–139.

[44] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijiver, *Combinatorial Optimization*. Hoboken, NJ: Wiley, 1998.

[45] T. R. Jensen and B. Toft, *Graph Coloring Problems*. New York: Wiley Interscience, 1995.

[46] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA: Kluwer, 1998.

[47] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Hoboken, NJ: Wiley, 1987.

Qingfu Zhang (M'01) received the B.Sc. degree in mathematics from Shanxi University, Shanxi, China, in 1984, and the M.Sc. degree in applied mathematics and the Ph.D. degree in information engineering from Xidian University, Xi'an, China, in 1991 and 1994, respectively.

From 1994 to 2000, he was with the National Laboratory of Parallel Processing and Computing, National University of Defence Science and Technology, China; the Hong Kong Polytechnic University, Hong Kong; the German National Research Center for Information Technology, Fraunhofer-Gesellschaft, Germany; and the University of Manchester Institute of Science and Technology, U.K. He is currently a Reader with the Department of Computer Science, University of Essex, Colchester, U.K. His main research areas are evolutionary computation, optimization, neural networks, data analysis, and their applications.

Dr. Zhang is an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B and a Guest Editor of a forthcoming special issue on evolutionary algorithms based on probabilistic models in the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

Jianyong Sun received the B.Sc. degree in mathematics from Xi'an Jiaotong University, Xi'an, China, in 1997, and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2005.

In 2000, he was a Research Assistant with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. From 2002 to 2003, he was a Senior Research Officer with the Department of Computer Science, University of Essex. He is currently a Research Associate with the School of Computer Science, University of Birmingham, Birmingham, U.K. His main research areas are evolutionary computation, optimization, theory and algorithms of metaheuristics, telecommunication networks, and machine learning.

Gaoxi Xiao (M'00) received the B.Sc. and M.Sc. degrees in applied mathematics from Xidian University, Xi'an, China, in 1991 and 1994, respectively, and the Ph.D. degree from the Hong Kong Polytechnic University, Kowloon, Hong Kong, in 1999.

From 1994 to 1995, he was with the Institute of Antenna and Electromagnetic Scattering, Xidian University. In 1999, he was a Postdoctoral Fellow with the Department of Electronic Engineering, Polytechnic University, Brooklyn, NY. In 1999–2001, he was a Visiting Scientist with the Center for Advanced Telecommunications Systems and Services, University of Texas, Dallas. Since October 2001, he has been an Assistant Professor with the Division of Communication Engineering, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include optical and wireless networking, complex networks, algorithm design and analysis, and network security.

Edward Tsang (M'04) received the B.S. degree in business administration from the Chinese University of Hong Kong, Hong Kong, in 1977 and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 1987.

He has broad interest in applied artificial intelligence, particularly in computational finance, scheduling, heuristic search, constraint satisfaction, and optimization. He is currently a Professor in computer science with the University of Essex, Colchester, U.K., where he leads the Computational Finance Group and the Constraint Satisfaction and Optimization Group. He is also the Deputy Director of the Centre for Computational Finance and Economic Agents, an interdisciplinary center.

Dr. Tsang chaired the Technical Committee for Computational Finance under the IEEE Computational Intelligence Society in 2004 and 2005.