Abstract—In this paper, we present a multi-objective Integer Linear Program (ILP) for the joint throughput optimization and traffic engineering problem in Wavelength-Division Multiplexing (WDM) networks. The proposed model simultaneously maximizes the aggregated throughput, minimizes the resource consumption, and achieves load balancing by minimizing the maximum link utilization (MLU). Even though the ILP is NP-hard, we demonstrate its application in a 14-node network. In addition to the ILP, we propose a heuristic algorithm which can be implemented in a distributed manner. The proposed algorithm addresses the multi-objective problem as an $e$-constraint problem, upper bounding the per-route resource consumption and maximizing the throughput by routing through multiple widest paths. Performance studies show that, by considering the multiple objectives simultaneously, the solutions of the ILP are better than those obtained by optimizing a single objective only. At the same time, the throughput of the heuristic is close to optimal, and load balancing is achieved. Furthermore, its resource consumption is nearly the same as the one obtained by the shortest path algorithm, which is optimal when only resource consumption is considered.

I. INTRODUCTION

Wavelength-Division Multiplexing (WDM) in optical fiber networks is the main technology used in tier-1 backbones. It provides the ability to carve up the huge optical bandwidth available in fiber into a lower-capacity non-interfering wavelengths or channels. In WDM networks, end-to-end connections must be routed such that no two connections use the same wavelength on a same link. Some networks additionally require a wavelength continuity constraint, where a connection must use the same wavelength along the path. However, the advent of wavelength converter devices may relax this constraint so that WDM networks become wavelength convertible.

Routing in WDM networks has been an active research area for several years. Much of the work is focused on maximizing the aggregated throughput in terms of accepted connections, which is equivalent to minimizing the blocking probability [1]. The latter is defined as the ratio of the rejected connections to the total number of connection requests. Other approaches consider traffic engineering, which attempts to balance the load of the network and to improve the resource consumption. Here, the latter refers to the total number of wavelengths used to route the aggregated connection requests.

Load balancing schemes mitigate hot spots and improve the reliability by routing through multiple paths, while minimizing resource consumption decreases the blocking probability of future connection requests. To the best of our knowledge, previous works considered either throughput optimization or traffic engineering, and optimized a single objective, while the other objectives were not under consideration.

In this paper, we present a multi-objective Integer Linear Program (ILP) which jointly considers the throughput optimization and the traffic engineering problems in wavelength convertible WDM networks. The proposed model simultaneously maximizes the aggregated throughput, minimizes the resource consumption, and achieves load balancing by minimizing the maximum link utilization (MLU). The latter metric measures the congestion of a network [2]. We also propose a heuristic algorithm which can be implemented in a distributed manner. The algorithm addresses the multi-objective problem as an $e$-constraint problem [3], upper bounding the per-route resource consumption and maximizing the throughput by routing through multiple widest paths.

The paper is organized as follows. Section II discusses related work. Section III formulates the joint throughput optimization and traffic engineering problem in WDM networks, and Section IV presents a heuristic algorithm to solve it. Section V shows performance studies, and Section VI concludes our work.

II. RELATED WORK

The throughput optimization problem in WDM networks can be seen as an extension of the multi-commodity flow problem in a graph, where flows and link capacities take integer values. Flows represent connection requests, while a link capacity represents the number of wavelengths available at the corresponding link. Banerjee et al. [1] presented an ILP that minimizes the average number packet hop distance. The authors claimed that optimizing the above objective is equivalent to maximizing throughput (i.e., the number of accepted connections). However, in general these two metrics may conflict with each other. Krishnaswamy et al. [4] provided two ILPs for the routing problem in WDM networks, and relaxed LP formulations with proper rounding techniques to

...
approximate to optimal solutions. Cavendish et al. [5] studied the routing and wavelength assignment (RWA) problem in WDM mesh networks with full wavelength conversion capabilities, and attempted to minimize multiple objectives such as the number of wavelength converters, the number of hops and the number of wavelengths used along the path; yet the authors just optimized one metric at a time. Belgaem et al. [6] proposed a heuristic to solve large instances of the throughput optimization problem in WDM networks. The heuristic partitions the problem into several smaller ILP subproblems that are optimally solved. Christodoulopoulos et al. [7] presented an ILP to optimize the number of wavelengths to satisfy certain connection requests, and a heuristic to solve it in a polynomial time. This problem is a dual of the throughput optimization; the goal is the minimization of the number of wavelengths instead of the maximization of the throughput. A traffic engineering scheme which minimizes congestion is proposed in [8]. To achieve this objective, traffic engineering schemes attempt to minimize the MLU.

III. JOINT THROUGHPUT OPTIMIZATION AND TRAFFIC ENGINEERING PROBLEM

We represent a wavelength convertible WDM network as a graph \( G = (V, E) \), where \( V \) is the set of nodes and \( E \) the set of links. Let \( N \) be the set of end-to-end flows or connection requests. Each flow is characterized by a 3-tuple \((s_n,d_n,r_n)\), which denotes the source node, the destination node and the number of connection requests. Let \( x_{ij}^n \) be a variable representing the number of connection requests of the \( n \)th flow routed on link \((i,j)\), and \( e_{ij} \) the capacity of the link \((i,j)\). Define \( f_n \leq r_n \), \( n \in N \), as the number of connection requests allocated to flow \( n \). Note that in typical throughput optimization problems, \( r_n \) is not given beforehand, and the objective is to maximize \( f_n \). The joint throughput optimization and traffic engineering problem is defined in Fig. 1.

\[
\text{Max } F = w_1 \cdot \sum_{n \in N} f_n - w_2 \cdot \sum_{n \in N} \sum_{(i,j) \in E} x_{ij}^n - w_3 \cdot \alpha
\]

\[
\sum_{j,(i,j) \in E} x_{ij}^n = \begin{cases} \sum_{j,(i,j) \in E} x_{ij}^n & \text{if } i = s_n; n \in N \ (1) \\ -f_n & \text{if } i = d_n; n \in N \ (1) \\ 0 & \text{otherwise} \end{cases}
\]

\[
\sum_{n \in N} x_{ij}^n \leq \alpha \cdot e_{ij}; (i,j) \in E
\]

\[
f_n \leq r_n; n \in N
\]

\[
x_{ij}^n \in \{0,1,2,...\}; n \in N, (i,j) \in E
\]

\[
f_n \in \{0,1,2,...,r_n\}; n \in N, (i,j) \in E
\]

\[
0 \leq \alpha \leq 1
\]

Fig. 1. Integer Linear Program 1 (ILP1).

The objective function \( F \) consists of three terms with their corresponding weights \( w_1, w_2 \) and \( w_3 \). The first term is the aggregated throughput. The second term is the resources (i.e., links) utilized by the scheme, and the third term, \( \alpha \), is the MLU. Minimizing the last two objectives is equivalent to maximizing the negative of them. Eq. (1) is the flow conservation constraint, and Eq. (2) is the link capacity constraint, where capacities are multiplied by the MLU. Eq. (3) restricts the number of connection requests allocated to flow \( n \) to be no more than its requested number of connections \( r_n \). This constraint permits us to upper bound the number of connections allocated to any flow \( n \). Eqs. (4) and (5) restrict the variables to be positive integers, and Eq. (6) states that the MLU is between 0 and 1. Even though ILP1 is NP-hard, it can be solved in a reasonable amount of time for small and medium sized networks by using branch and bound techniques. ILP1 is a multi-objective problem, where the objective function \( F \) may be expressed as an objective vector \( F = (F_1, F_2, F_3) \), where:

\[
F_1 = \sum_{n \in N} f_n \text{ (throughput) } \quad(7)
\]

\[
F_2 = -\sum_{n \in N} \sum_{(i,j) \in E} x_{ij}^n \text{ (resource consumption) } \quad(8)
\]

\[
F_3 = -\alpha \text{ (MLU) } \quad(9)
\]

The relative importance of each objective can be varied according to the weight vector \( \bar{w} = (w_1, w_2, w_3) \). We will consider the routing of all connection requests as the main goal. Thus, we will set \( w_1 >> w_2 \) and \( w_1 >> w_3 \).

In next sections, we will use Pareto optimality concepts [3]. Let \( \bar{w}(a), \bar{w}(b) \) be two weight vectors, and \( F(a), F(b) \) be the objective vectors corresponding to the solutions of ILP1 with \( \bar{w}(a) \) and \( \bar{w}(b) \) respectively. We will say that the solution obtained by setting ILP1 with \( \bar{w}(a) \) is better than that with \( \bar{w}(b) \) if and only if (iff) \( F_i(a) > F_i(b) \) for some \( i \in \{1,2,3\} \), and \( F_i(a) \geq F_i(b) \) for the remainder objectives. We denote this relation as \( F(a) > F(b) \).

IV. PROPOSED HEURISTIC

The proposed algorithm consists of two steps: i) modify the original graph to a hop count constrained one; and ii) apply the Ford-Fulkerson method, routing connection requests along the widest path in the modified graph [9][pp. 199-230]. The widest path is defined as the path with maximum capacity. The capacity of a path \( P \) is defined as \( C(P) = \min_{(i,j) \in P} e_{ij}' \), where \( e_{ij}' \) refers to the residual link capacity (i.e., available number of wavelengths) of link \((i,j)\). By routing through widest paths, the algorithm attempts to optimize not only throughput but also to achieve load balancing.

A. Step 1: Graph Conversion

The first step of the algorithm constraints the resources used by a connection request (i.e., it upper bounds the resources used for any path in the graph). The procedure to modify the graph is based on the heuristic proposed by Seok et al. [2]. For a given flow \( n \in N \) and the network \( G = (V, E) \), the hop count constrained graph \( G' = (V', E') \) guarantees that any path from \( s_n \) to \( d_n \) has no more than \( H_n \) hops or links. \( V' \)

\[
H_n = H + H^\text{min}_n, \text{ where } H^\text{min}_n \text{ is the number of hops of the shortest path from } s_n \text{ to } d_n. \text{ Thus, } H \text{ can be considered as a relaxation in the number of hops or links (i.e., resources) per-route that can be allocated to flow } n.
\]

\[
1 H_n = H + H^\text{min}_n, \text{ where } H^\text{min}_n \text{ is the number of hops of the shortest path from } s_n \text{ to } d_n. \text{ Thus, } H \text{ can be considered as a relaxation in the number of hops or links (i.e., resources) per-route that can be allocated to flow } n.
\]
Fig. 2. Graph conversion example. In (a), the numbers over each link refer to the current number of wavelengths used and the total number of wavelengths of the corresponding link. In (b), the number of hops of any path from node 0 to node 3 is limited to 2. The number over each link is the number of wavelengths available.

and $E'$ are given as:

$$V' = \cup_{0 \leq k \leq H_n} V'_k, V'_0 = \{s_n\}, \forall i \in E, i \in V'_{k-1},$$

$$E' = \cup_{0 \leq k \leq H_n} E'_k, E'_0 = \{(s_n, j) | (s_n, j) \in E\},$$

$$E'_k = \{(i, j) | i \in E'_{k-1}, j \in E'_k, (i, j) \in E\}.$$

Fig. 2 shows an example of a modified graph for a connection from node 0 to node 3, and a hop constraint $H_0 = 2$. The numbers over each link in Fig. 2(a) represent the number of wavelengths currently used and the total number of wavelengths of the link; in Fig. 2(b), only the residual capacity is relevant.

The graph conversion attempts to alleviate the difficulty faced by the simultaneous optimization of multiple metrics, treating the resource consumption objective as a restriction (i.e., the per-route resource consumption is limited by $H_0$). This method of addressing multi-objective problems is known as $\epsilon$-constraint technique [3](pp. 57-60).

B. Step 2: Ford-Fulkerson-Widest-Path

The second step consists of the Ford-Fulkerson method with widest path algorithm (FFWP), which is shown in Fig. 3. It attempts to allocate $r$ connection requests from node $s$ to node $d$ by successively finding widest paths, until the number of connection requests $r$ already allocated is equal to $r$. The procedure Widest-Path shown in Fig. 4 is based on the Dijkstra’s algorithm. The capacity or cost $C(u)$ of a node $u$ in line 2 represents the capacity of the widest path from the source node $s$ to the node $u$. The predecessor $\text{pred}(u)$ refers to the predecessor of node $u$ in the widest path to $u$, as in a typical Dijkstra’s implementation. The algorithm uses a max-priority queue $Q$ for maintaining the set of nodes $V$ and its associated capacities $C(u), \forall u \in V$. The queue uses the following standard operations: MakeMaxQueue($V$), DeleteMax($Q$), and IncreaseKey($Q, v, C(v)$). MakeMaxQueue($V$) makes a max-priority queue for the set of nodes $V$; DeleteMax($Q$) returns the element in $Q$ with the largest capacity; and IncreaseKey($Q, v, C(v)$) increases the capacity of node $v$ to the new value $C(v)$ [10](pp. 138-144).

To illustrate the operation of FFWP, consider again the example of Fig. 2(b) and assume a connection request $r = 3$. The algorithm routes the 3 connections through node 2 (because the widest path is $(0, 2), (2, 3)$). For $r = 4$, the algorithm routes 3 connections through node 2, and then 1 connection through node 1 (because the next widest path is $(0, 1), (1, 3)$).

C. Complexity Analysis

Step 1 can be computed offline. Thus, we will consider the running time of step 2 only. In each iteration of FFWP, the algorithm finds the widest path $P$ in the residual network3, and allocates $r'_i$ connections through this path (Fig. 3, line 9). Let $K$ be the number of paths found by FFWP, and $l_i$ be the number of connection requests still not routed after $i$ iterations ($l_0 = r$). At the first iteration, FFWP finds the widest path $P$, which has a capacity $C(P) \geq \left\lceil \frac{l_0}{K} \right\rceil$. Thus, at least $\left\lceil \frac{l_0}{K} \right\rceil$ connections are routed through this path, and $l_1 \leq l_0 - \left\lceil \frac{l_0}{K} \right\rceil$.

At an iteration $i$, the residual graph must have a path with capacity of at least $\left\lceil \frac{l_{i-1}}{K} \right\rceil$; otherwise, it would not be possible to route $l_{i-1}$ connections on that residual graph. The number

![Fig. 3. Ford-Fulkerson-Widest-Path (FFWP) algorithm. The input parameters are the graph $G'$ (which was found in Step 1), the source and destination nodes $s$ and $d$, and the number of connection requests $r$. The variables $r_a$ and $r'_a$ represent the total number of connection requests already allocated and the number of connection requests allocated during the current iteration.

<table>
<thead>
<tr>
<th>Widest-Path($G$, $s, d$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: for all $u \in V$ do</td>
</tr>
<tr>
<td>2: $C(u) = 0$;</td>
</tr>
<tr>
<td>3: $\text{pred}(u) = \text{NULL}$;</td>
</tr>
<tr>
<td>4: end for</td>
</tr>
<tr>
<td>5: $C(s) = \infty$;</td>
</tr>
<tr>
<td>6: $Q = \text{MakeMaxQueue}(V)$;</td>
</tr>
<tr>
<td>7: while ($u = \text{DeleteMax}(Q) \neq d$) do</td>
</tr>
<tr>
<td>8: for all $(u, v) \in E$ do</td>
</tr>
<tr>
<td>9: $\text{cap} = \min(C(u), c_{uv})$;</td>
</tr>
<tr>
<td>10: if $\text{cap} &gt; C(v)$ then</td>
</tr>
<tr>
<td>11: $C(v) = \text{cap}$;</td>
</tr>
<tr>
<td>12: $\text{pred}(v) = u$;</td>
</tr>
<tr>
<td>13: IncreaseKey($Q, v, C(v)$);</td>
</tr>
<tr>
<td>14: end if</td>
</tr>
<tr>
<td>15: end for</td>
</tr>
<tr>
<td>16: end while</td>
</tr>
<tr>
<td>17: if $C(d) &gt; 0$ then</td>
</tr>
<tr>
<td>18: $P = \text{Path given by predecessors}$;</td>
</tr>
<tr>
<td>19: else</td>
</tr>
<tr>
<td>20: $P = \text{NULL}$;</td>
</tr>
<tr>
<td>21: end if</td>
</tr>
<tr>
<td>22: return $P$;</td>
</tr>
</tbody>
</table>

![Fig. 4. Widest Path algorithm. It returns the widest path from node $s$ to node $d$ on the graph $G$.](1045)
of connection requests still not routed after iteration \( i \) is:

\[
\begin{align*}
  l_i &\leq l_{i-1} - \frac{l_{i-1}}{K} \leq (a) l_{i-1} \left( 1 - \frac{1}{K} \right) \\
  &\leq (b) l_0 \left( 1 - \frac{1}{K} \right)^i \leq (c) l_0 e^{-\frac{i}{K}} = re^{-\frac{i}{K}},
\end{align*}
\]

where (a) follows from bounding the negative of the floor function and then factoring \( l_{i-1} \), (b) by repeated application of the previous recurrence relation, and (c) by applying the inequality \( 1 - x \leq e^{-x} \), for all \( x \), with equality iff \( x = 0 \). The term \( re^{-\frac{i}{K}} \) is equal to 1 when \( i = K \log_r r \). If the network has no capacity to route \( r \) connection requests, the number of iteration is smaller than \( K \log_r r \). Given that \( K \leq |E| \), the total number of iterations is \( O(|E| \log |V|) \). Since the running time of the widest path algorithm shown in Fig. 4 is \( O(|E|^2 \log r \log |V|) \) when implemented with a binary heap, the total running time is \( O(|E|^2 \log r \log |V|) \).

V. PERFORMANCE STUDIES

We first present an example of ILP1, which illustrates the impact of the weight vector \( \vec{w} \). We solved ILP1 with three different weight vectors: \( \vec{w}(a) = (1, 0, 0) \), \( \vec{w}(b) = (0.9, 0.1, 0) \), and \( \vec{w}(c) = (0.9, 0.05, 0.05) \). With the first weight vector, we study the case when \( F_1 \) only is optimized. Then, we analyze the impact of slightly increasing \( w_2 \) such that the resource consumption is also considered. Finally, we optimize the three objectives: by setting \( w_1 >> w_2 > 0 \) and \( w_1 >> w_3 > 0 \), the the resource consumption and MLU are minimized without compromising throughput. Consider the NSF network topology shown in Fig. 6, and assume that \( c_{ij} = 10, \forall (i, j) \in E \) (i.e., each link has 10 wavelengths). We generated 7 random flows with the same number of connection requests per flow, namely \( (0, 14), (3, 13), (1, 12), (13, 1), (12, 0), (15, 5), (9, 4) \). Note that with these source-destination pairs, flows flow extensively across the network. Fig. 5(a) shows the blocking probability as the number of connection requests increases. The blocking probability \( p_o \) is a metric directly related to the throughput given by Eq. (7). Specifically, \( p_o = 1 - \frac{F_1}{P_2} c_{\text{OPT}} \), which implies that the maximization of \( F_1 \) leads to the minimization of the blocking probability. The results regarding this metric are the same for the three weight vectors, because \( w_1 >> w_2 \) and \( w_1 >> w_3 \) in all cases. Fig. 5(b) shows the resource consumption normalized to the total network resources, i.e., \( \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} \). We can see that the resources consumed by \( \vec{w}(b) \) and \( \vec{w}(c) \) are lower than those consumed by \( \vec{w}(a) \), which may improve the blocking probability of future connection requests. Fig. 5(c) shows that better load balancing can be achieved by increasing weight \( w_3 \). Considering the three objectives simultaneously, \( \vec{F}(c) > \vec{F}(b) > \vec{F}(a) \). Thus, by simultaneously optimizing multiple objectives and appropriately setting \( \vec{w} \), better solutions than a traditional single objective approach can be obtained.

Consider again the same set of flows and network topology. Fig. 7 shows the results obtained by solving ILP1 with \( \vec{w} = (0.9, 0.05, 0.05) \), and the results obtained with the heuristic proposed in Section IV, with \( H = 0 \) and \( H = 2 \). For reference purposes, we also include the results obtained with the shortest path (SP) algorithm. As expected, the blocking probability of SP quickly increases with the number of connection requests. For \( H = 0 \), the blocking probability achieved by the heuristic is 0 until the number of connection requests reaches 35, and then monotonically increases. By relaxing \( H \) to 2, the blocking probability improves and is 0 until the number of connection requests increases to 70. Define the blocking probability surcharge \( \delta \) of an algorithm as the gap between its blocking probability and the optimal blocking probability computed by ILP1, i.e., \( \delta = \frac{F_1}{F_2} \). Fig. 7(b) shows that the gap between the optimal and the heuristic converges to 14% as the number of connection requests increases. On the other hand, the shortest path shows a gap of up to 43%. Fig. 7(c) shows the results in terms of MLU. The SP has a very high link utilization, even when the number of connection requests is only 7. The MLU produced by the ILP is better than the MLU achieved by heuristic with \( H = 0 \), but higher than the MLU...
obtained by the heuristic with $H = 2$. ILP1 produced a higher MLU because of the small value of $w_3$. This implies that $\vec{w} = (0.9, 0.05, 0.05)$ may not be appropriately set, and better results in terms of MLU may be achieved by carefully setting the weight vector. Fig. 7(d) shows the resource consumption. SP achieves the lowest consumption, since it routes all the connection requests by using the minimum number of links. However, as seen in Figs. 7(a)-(c), the impact of this strategy is a poor performance in terms of blocking probability and MLU. The performance of the heuristic regarding resource consumption is very close to the SP, because of the constraint imposed by $H$. On the other hand, ILP1 produces a higher consumption, since the weight vector gives preference to the throughput objective. However, better solutions in terms of resource consumption (without compromising throughput) may be also obtained by appropriately setting $\vec{w}$.

VI. CONCLUSION

In this paper, we have presented a multi-objective ILP for the joint throughput optimization and traffic engineering problem in WDM networks. The proposed model simultaneously maximizes the aggregated throughput, minimizes the resource consumption, and achieves load balancing by minimizing the maximum link utilization (MLU). Performance studies show that, by considering the multiple objectives simultaneously, better solutions can be obtained. However, further studies are needed to appropriately set the weight vector. We have also proposed a heuristic algorithm which can be implemented in a distributed manner. The algorithm addresses the joint multi-objective throughput optimization and traffic engineering problem as an $\epsilon$-constraint problem, upper bounding the per-route resource consumption and routing the connection requests through multiple widest paths. Its performance is close to optimal regarding blocking probability, while at the same time it achieves low MLU. Furthermore, its resource consumption is nearly the same as the one obtained by SP, which is optimal when only resource consumption is considered. Future work includes the application of the proposed ILP and heuristic to real Internet traffic traces. We will also develop a systematic procedure to determine the appropriate weight vector.

REFERENCES