An Experimental Study of Maximum Profit Wavelength Assignment in WDM Rings

Evangelos Bampas, Aris Pagourtzis, and Katerina Potika

School of Elec. & Comp. Engineering, National Technical University of Athens
{ebamp,pagour,epotik}@cs.ntua.gr

Abstract. We are interested in the problem of satisfying a maximum-profit subset of undirected communication requests in an optical ring that employs the Wavelength Division Multiplexing technology. We present four deterministic and purely combinatorial algorithms for this problem, and give theoretical guarantees for their worst-case approximation ratios. Two of these algorithms are novel, while the rest are adaptation of earlier approaches. An experimental evaluation of the algorithms in terms of attained profit and execution time reveals that the theoretically best algorithm performs only marginally better than one of the new algorithms, while at the same time being several orders of magnitude slower. Furthermore, an extremely fast greedy heuristic with non-constant approximation ratio performs reasonably well and may be favored over the other algorithms whenever it is crucial to minimize execution time.

1 Introduction

Wavelength Division Multiplexing (WDM) is a dominating technology in contemporary all-optical networking. It allows several connections to be established through the same fiber links, provided that each of the connections uses a different wavelength. A second requirement is that a connection must use the same wavelength from one end to the other in order to avoid the use of wavelength converters which are costly or slow. In practice, the available bandwidth is limited to a few dozen, or at most hundred, wavelengths per fiber and the situation is not expected to change in the near future. It is therefore impossible to serve a large enough set of communication requests simultaneously. It thus makes sense to consider the problem of satisfying a maximum-profit subset of requests, where profits may represent priorities or actual revenues associated with the communication requests.

In our model, requests are undirected and pre-routed. Undirected requests correspond to *full-duplex* communication. In this mode of communication, it is assumed that each physical link in the network is implemented with two parallel optical fibers. Each fiber is reserved for carrying data in one direction only. Whenever a request between two nodes is assigned a wavelength, this wavelength is reserved for this request on *both* parallel paths connecting the two nodes; each path is used for transferring data in one direction only. Two requests or paths are assumed to *overlap* when they share a physical link of the network. Moreover,

pre-routed requests arise in settings where the path on which a request will be routed is decided independently of the wavelength assignment procedure. This is the case when there are specific routing requirements, such as shortest-path routing, or when lightpaths are set up in an earlier stage of the virtual topology design process.

In the wavelength assignment problem that we study, we are given a set of simple paths in a network (representing full-duplex communication requests), a profit associated with each path, and a number of available wavelengths. A solution is any subset of the paths that can be assigned wavelengths in such a way that overlapping paths receive different wavelengths. The goal is to find a solution that maximizes the total profit of satisfied requests. We call this problem Maximum Profit Path Coloring, or MaxProfit-PC for short. We will focus on the MaxProfit-PC problem where the underlying network is a ring, which is a fundamental network topology and is frequently deployed in practice (for example, in the case of SONET rings–Synchronous Optical Network rings). Note, also, that MaxProfit-PC captures request satisfaction problems in any kind of networks if we interpret colors as time slots. Then, MaxProfit-PC corresponds to the problem of maximizing the profit of requests that can be satisfied in a given number of time slots.

1.1 Related Work

While the cardinality version of the problem (called MaxPC, for Maximum Path Coloring) has been studied by several researchers [1–5], MaxProfit-PC has been considered in rather few papers [4, 5]. Both MaxProfit-PC and MaxPC are NP-hard even in simple networks such as rings and trees; this can be shown by an immediate reduction from the corresponding color minimization problem (see e.g. [1]).

MaxProfit-PC in chains is also known as the "weighted k-coloring of intervals" problem, which can be solved exactly as shown by Carlisle and Lloyd [5]. In the case of MaxProfit-PC in rings, Caragiannis [4] has presented a randomized algorithm based on linear programming that achieves an expected approximation ratio of 0.67. Let us note here that, although the algorithm in [4] achieves a slightly better worst-case approximation ratio than one of our algorithms, we have chosen not to include it in our experimental comparison, since our focus is on deterministic and purely combinatorial algorithms.

MaxPC in rings admits a $\frac{3}{4}$-approximation algorithm, as shown by Caragiannis [4]. Wan and Liu [1] present a $(1 - \frac{1}{e})$-approximation algorithm for MaxPC in trees. In [1] they also study the case where requests are not routed in advance and present a constant approximation algorithm for meshes, as well as a $(1 - \frac{1}{e})$-approximation algorithm for rings. The latter was recently superseded by an improved $\frac{2}{3}$-approximation algorithm due to Bian and Gu [6]. Li et al. [7] also study a version of MaxProfit-PC where requests are not routed in advance; in this version they assume directed requests and edge capacities that must be obeyed, and present a $\frac{1}{2}$-approximation algorithm for rings.

1.2 Our Results

In this paper, we present four algorithms for MaxProfit-PC in rings with undirected requests. Our algorithms combine ideas from algorithms for MaxPC [3, 1] with new techniques specially designed for coloring paths with profits. We give theoretical bounds on the approximation ratio achieved by these algorithms and then move on to perform an experimental comparison with respect to the total profit of the solutions they produce and the execution time they require.

One of the results of this comparison is that Match-and-Replace, a novel algorithm that we propose, performs only marginally worse than Iterative, which is based on a well-known technique and gives the best theoretical guarantee for the approximation ratio among the implemented algorithms. At the same time, Match-and-Replace is several orders of magnitude faster than Iterative. A second finding of the experimental comparison is that a natural greedy heuristic with non-constant theoretical approximation guarantee actually performs quite competently and is also exceptionally fast.

The rest of the paper is organized as follows: In Section 2 we give a few preliminary definitions. In Section 3 we present in detail the Match-and-Replace algorithm. In Section 4 we present in detail the other three algorithms to be included in the comparison. In Section 5 we describe the experimental setup that we used and discuss the numerical results. We conclude in Section 6 with a ranking of the algorithms with respect to their performance in the experiments.

2 Preliminaries

Let $w : \mathcal{P} \to \mathbb{Q}^+$ be a function assigning weights to the paths in some set $\mathcal{P}$. For any $A \subseteq \mathcal{P}$, we will employ the notation $w(A)$ for the total weight of A: $w(A) = \sum_{p \in A} w(p)$. Similarly, for any set $\mathcal{S}$ of subsets of $\mathcal{P}$, we will employ the notation $w(\mathcal{S})$ for the sum of total weights of the elements of $\mathcal{S}$: $w(\mathcal{S}) = \sum_{A \in \mathcal{S}} w(A)$. Note that, if $\mathcal{S}$ contains mutually disjoint subsets of $\mathcal{P}$, then $w(\mathcal{S}) = w\left(\bigcup_{A \in \mathcal{S}} A\right)$.

We formally define the MaxProfit-PC problem in graph-theoretic terms as follows:

Definition 1. Maximum Profit Path Coloring Problem (MaxProfit-PC)
Input*: $\langle G, \mathcal{P}, w, k \rangle$, where G is an undirected graph, $\mathcal{P}$ is a set of simple paths defined on G, $w : \mathcal{P} \to \mathbb{Q}^+$ is a profit function, and $k \in \mathbb{N}^+$ is the number of available colors.*
Feasible solution*: a set of paths $\mathcal{P}' \subseteq \mathcal{P}$ that can be colored with k colors so that no overlapping paths are assigned the same color.*
Goal*: maximize $w(\mathcal{P}')$.*

We also define MaxPC, which is the cardinality version of MaxProfit-PC (equivalent to all paths having profit equal to 1).

Definition 2. Maximum Path Coloring Problem (MaxPC)
Input*: $\langle G, \mathcal{P}, k \rangle$, where G is an undirected graph, $\mathcal{P}$ is a set of simple paths defined on G, and $k \in \mathbb{N}^+$ is the number of available colors.*
Feasible solution*: a set of paths $\mathcal{P}' \subseteq \mathcal{P}$ that can be colored with k colors so that no overlapping paths are assigned the same color.*
Goal*: maximize $|\mathcal{P}'|$.*

Note that, in general, there may be multiple paths defined on the same set of edges of a graph. We assume that each path in a given instance of the problem is distinguished by a unique ID and thus we speak of *sets* instead of *multisets* of paths. We will not make explicit use of path ID's hereafter.

A *chain* is a graph that consists of a single path, while a *ring* is a graph that consists of a single cycle. If we remove an edge e from a ring we get a chain; we call such an edge a *separation edge*.

Given a network $G = (V, E)$ and a set of paths $\mathcal{P}$, we denote by n the size of set V, and by m the size of set $\mathcal{P}$. If $\mathcal{P}$ is a set of paths and e is an edge, then we denote by $L(e, \mathcal{P})$ the *load of edge e with respect to $\mathcal{P}$*, i.e. the number of paths in $\mathcal{P}$ that use e. We denote by $length(p)$ the number of edges of path p. Given a set of paths $\mathcal{P}$ and a coloring thereof, the subset of $\mathcal{P}$ that is colored with color i is called the *i-th color class* of $\mathcal{P}$ and is denoted by $\mathcal{P}(i)$.

Carlisle and Lloyd [5] give an exact algorithm for MaxProfit-PC in chains that runs in $O(km \log m)$ time. In the sequel, we will often use this algorithm as a subroutine for the algorithms that we present. We will refer to this algorithm as the Carlisle-Lloyd Algorithm.

An algorithm A for a maximization problem Π is a *ρ-approximation algorithm* (for $0 < \rho \leq 1$) if for every instance I of Π, A runs in time polynomial in $|I|$ and delivers a solution with profit $SOL \geq \rho \cdot OPT$, where OPT denotes the profit of an optimal solution for I.

3 The Match-and-Replace Algorithm

We propose a novel algorithm for MaxProfit-PC in rings. The Match-and-Replace algorithm is based on a popular technique used for rings, namely to pick a separation edge and remove it from the ring. The set of requests is then partitioned, with respect to the separation edge, into two subsets: the subset of requests that use the separation edge, and the subset of requests that do not use it. Observe that the latter subset can be regarded as an instance of MaxProfit-PC in a chain, and thus it can be colored optimally in polynomial time. After this step, the algorithm tries to color some of the requests that use the separation edge, possibly sacrificing some of the requests that have already been colored. To that end, it computes a maximum-weight matching on the corresponding *compatibility graph*:

Definition 3 (Compatibility graph). *Let $\langle G = (V, E), \mathcal{P}, w, k \rangle$ be an instance of* MaxProfit-PC *and $e \in E$ be a separation edge that partitions the path set $\mathcal{P}$ into $\mathcal{P}_e$ (paths in $\mathcal{P}$ that use edge e) and $\mathcal{P}_c = \mathcal{P} \setminus \mathcal{P}_e$. For*

Algorithm 1 Match-and-Replace

Input: an instance $\langle G, \mathcal{P}, w, k\rangle$ of MaxProfit-PC, where G is a ring

1: Pick an arbitrary separation edge e of the ring. Let $\mathcal{P}_e$ be the set of paths that use edge e and $\mathcal{P}_c = \mathcal{P} \setminus \mathcal{P}_e$.
2: Color the instance $\langle G - e, \mathcal{P}_c, w, k\rangle$ optimally, using the Carlisle-Lloyd Algorithm for MaxProfit-PC in chains.
3: Let $\mathcal{P}_c(i)$ be the i-th color class of $\mathcal{P}_c, 1 \leq i \leq k$ (note that some color classes may be empty).
4: Construct the compatibility graph $\mathcal{H}$ that corresponds to the separation edge picked in Step 1 and the partial coloring obtained in Step 3.
5: Find a maximum-weight matching M of $\mathcal{H}$.
6: **for each** edge $(\mathcal{P}_c(i), q) \in M$ **do**
7: uncolor all paths in $\mathcal{P}_c(i)^q$ and color path $q \in \mathcal{P}_e$ with color i.
8: **end for**

any partial coloring of the paths in $\mathcal{P}_c$, we define the compatibility graph $\mathcal{H}$ *as follows: $\mathcal{H} = (U, D)$ is a weighted complete bipartite graph with node set $U = \{\mathcal{P}_c(i) : 1 \leq i \leq k\} \cup \mathcal{P}_e$ and edge weights*

$$h(\mathcal{P}_c(i), q) = w(q) - w(\mathcal{P}_c(i)^q) \ ,$$

where $\mathcal{P}_c(i)^q$ is the subset of $\mathcal{P}_c(i)$ that contains all paths that overlap with q.

A detailed description of the algorithm is presented in Algorithm 1. We prove below that this algorithm achieves an approximation ratio of $\frac{1}{2}$, and that the analysis that we provide is tight.

Theorem 1. *The* Match-and-Replace *algorithm achieves an approximation ratio of $\frac{1}{2}$ for* MaxProfit-PC *in rings.*

Proof. Let OPT be the value of any optimal solution of the ring instance, OPT_c be the value of any optimal solution of the instance constrained to path set $\mathcal{P}_c$, and OPT_e be the value of any optimal solution of the instance constrained to path set $\mathcal{P}_e$. Because $\mathcal{P}_e$ and $\mathcal{P}_c$ form a partition of $\mathcal{P}$,

$$OPT \leq OPT_c + OPT_e \ . \tag{1}$$

Let SOL_c be the value of the solution obtained in Step 2 of the algorithm (chain subinstance solution), and SOL be the value of the final solution. Clearly,

$$SOL = SOL_c + h(M) \tag{2}$$

where $h(M)$ is the sum of the weights of the edges that belong to the matching M computed in Step 5 (recall that h is the edge weight function of the compatibility graph $\mathcal{H}$). The instance $\langle G - e, \mathcal{P}_c, w, k\rangle$ is solved optimally in Step 2. Therefore, taking also into account Equation 2 we have that:

$$OPT_c = SOL_c \leq SOL \ . \tag{3}$$

Let $\mathcal{S} = \{\mathcal{P}_c(i) : 1 \leq i \leq k\}$, and $\mathcal{S}_M$ be the set of $\mathcal{P}_c(i)$'s that are matched by M. Similarly, let $\mathcal{P}_{e,M}$ be the paths in $\mathcal{P}_e$ that are matched by M. Finally, let K be the set of the k most profitable paths of $\mathcal{P}_e$. We will now show that

$$OPT_e = w(K) \leq SOL \ . \tag{4}$$

For the sake of analysis we will examine a solution SOL' that Match-and-Replace would have computed if it had chosen a matching M' of a subgraph $\mathcal{H}'$ of $\mathcal{H}$ in Step 5. The bipartite graph $\mathcal{H}'$ has the same node set and the same edge weight function as $\mathcal{H}$, but only a subset of the edges of $\mathcal{H}$. More specifically, for every pair $(\mathcal{P}_c(i), q)$: the edge $(\mathcal{P}_c(i), q)$ is in $\mathcal{H}'$, if $w(q) - w(\mathcal{P}_c(i)) > 0$ and $q \in K$. Let M' be a maximum matching in $\mathcal{H}'$, and let $\mathcal{S}_{M'}$ and $\mathcal{P}_{e,M'}$ be defined analogously for M' as for M. Similarly to Equation 2,

$$SOL' = SOL_c + h(M') \ . \tag{5}$$

Note that $SOL_c = w(\mathcal{S})$. Denoting by $\mathcal{P}_c(i)^{\neg q}$ the set of paths in $\mathcal{P}_c(i)$ that *do not* overlap with q, we have:

$$\begin{aligned} h(M') &= w(\mathcal{P}_{e,M'}) - \sum_{(P(i),q)\in M'} w(\mathcal{P}_c(i)^q) \\ &= w(\mathcal{P}_{e,M'}) - \sum_{(\mathcal{P}_c(i),q)\in M'} [w(\mathcal{P}_c(i)) - w(\mathcal{P}_c(i)^{\neg q})] \\ &= w(\mathcal{P}_{e,M'}) - w(\mathcal{S}_{M'}) + \sum_{(\mathcal{P}_c(i),q)\in M'} w(\mathcal{P}_c(i)^{\neg q}) \ , \end{aligned}$$

Equation 5 may then be rewritten as follows:

$$SOL' = w(\mathcal{S} \setminus \mathcal{S}_{M'}) + w(\mathcal{P}_{e,M'}) + \sum_{(\mathcal{P}_c(i),q)\in M'} w(\mathcal{P}_c(i)^{\neg q}) \ .$$

We observe that $\mathcal{P}_{e,M'} \subseteq K$ and therefore $w(\mathcal{P}_{e,M'}) + w(K \setminus \mathcal{P}_{e,M'}) = w(K)$, so the last sum can be expanded in the following way:

$$SOL' = w(\mathcal{S} \setminus \mathcal{S}_{M'}) + w(K) - w(K \setminus \mathcal{P}_{e,M'}) + \sum_{(\mathcal{P}_c(i),q)\in M'} w(\mathcal{P}_c(i)^{\neg q}) \ . \tag{6}$$

Observe also that for any $\mathcal{P}_c(i) \notin \mathcal{S}_{M'}$ and $q \notin \mathcal{P}_{e,M'}$, there must be no edge between them in $\mathcal{H}'$, hence $w(\mathcal{P}_c(i)) \geq w(q)$. Moreover, $w(\mathcal{S} \setminus \mathcal{S}_{M'})$ and $w(K \setminus \mathcal{P}_{e,M'})$ are sums with the same number of terms because $|K| = |\mathcal{S}| = k$ and $|\mathcal{S}_{M'}| = |\mathcal{P}_{e,M'}|$. These observations imply that $w(\mathcal{S} \setminus \mathcal{S}_{M'}) - w(K \setminus \mathcal{P}_{e,M}) \geq 0$, therefore Equation 6 yields:

$$SOL' \geq w(K) \ . \tag{7}$$

Since $\mathcal{H}'$ is a subgraph of $\mathcal{H}$, M' is a matching also for $\mathcal{H}$, although probably not a maximum-weight one. Therefore, $h(M) \geq h(M')$, which implies, from

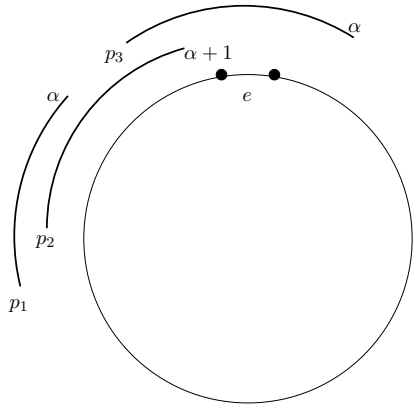

Fig. 1. An instance of MaxProfit-PC in which the Match-and-Replace algorithm performs as badly as possible. There is only one available color and three paths, p_1, p_2, and p_3 with profits α, $\alpha+1$, and α respectively. Assuming that Match-and-Replace picks edge e as separation edge in Step 1, it will color path p_2 for a profit of $\alpha+1$, while the optimal solution would be to color paths p_1 and p_3 for a profit of 2α. The value of α is arbitrary.

Equations 2 and 5, that $SOL \geq SOL'$. Combining this last inequality with Equation 7, we obtain Equation 4.

By Equations 3 and 4, SOL is an upper bound on both OPT_e and OPT_c, which together with Equation 1 gives:

$$SOL \geq \frac{OPT}{2} \ .$$

□

Example 1 (Tight example for the approximation ratio of Match-and-Replace*).* Consider the MaxProfit-PC instance illustrated in Figure 1. There is only one available color and three paths p_1, p_2, and p_3. Paths p_1 and p_3 are non-overlapping, while p_2 overlaps with both p_1 and p_3. The profits of the paths are: $w(p_1) = w(p_3) = \alpha$ and $w(p_2) = \alpha + 1$, where α is an arbitrary value. Assuming that edge e, as shown in Figure 1, is picked as separation edge in Step 1, it is straightforward to verify that Match-and-Replace will color path p_2 with the only available color, while the optimal solution would be to color paths p_1 and p_3. Therefore, the profit of the solution returned by the algorithm can be as bad as a fraction $\frac{\alpha+1}{2\alpha}$ of the optimal, which approaches $\frac{1}{2}$ as α goes to infinity.

Time complexity of Match-and-Replace*:* The most time-consuming step of the algorithm is Step 5. The compatibility graph $\mathcal{H}$ has $O(k+m)$ nodes and $O(km)$ edges; recall that m is the number of paths and k is the number of available colors in the original instance. The maximum-weight matching computation of

Algorithm 2 Best Choice

Input: an instance $\langle G, \mathcal{P}, w, k\rangle$ of MaxProfit-PC, where G is a ring

1: Pick an arbitrary separation edge e of the ring. Let $\mathcal{P}_e$ be the set of paths that use edge e and $\mathcal{P}_c = \mathcal{P} \setminus \mathcal{P}_e$.
2: Color the instance $\langle G - e, \mathcal{P}_c, w, k\rangle$ optimally, using the Carlisle-Lloyd Algorithm for MaxProfit-PC in chains. Let k' be the number of colors used in this partial coloring. Use the remaining colors, if any, for the $k - k'$ most profitable paths in $\mathcal{P}_e$. Let $\mathcal{P}_A$ be the set of colored paths.
3: Let $\mathcal{P}_B$ be the set of the k most profitable paths in $\mathcal{P}_e$.
4: **if** $w(\mathcal{P}_A) > w(\mathcal{P}_B)$ **then** return the coloring obtained in Step 2 for $\mathcal{P}_A$.
5: **else** return the coloring that uses a different color for each path in $\mathcal{P}_B$.
6: **end if**

Step 5 requires, therefore, $O(km(k+m) + (k+m)^2 \log(k+m))$ time. Under the reasonable assumption that $k \ll m$, the time complexity of the algorithm becomes $O(m^2(k + \log m))$.

4 Other Approaches for Approximating MaxProfit-PC

In this section we present three more algorithms for MaxProfit-PC, which we call Best Choice, Iterative, and MPLU-Greedy.

4.1 Best Choice

A second, more naive application of the separation edge technique involves picking the best of the following two solutions:

1. the solution obtained by coloring optimally the paths that do not use the separation edge, and
2. using one color for each of the k most profitable paths that use the separation edge.

We call this algorithm Best Choice and a detailed description is given in Algorithm 2. This algorithm also achieves an approximation ratio of $\frac{1}{2}$ for MaxProfit-PC in rings: the upper bound on OPT given by Equation 1 holds also for the separation edge e picked in Step 1; besides, the profits of the solutions obtained in Steps 2 and 3 are at least as large as OPT_c and OPT_e, respectively. We thus have:

Theorem 2. *The* Best Choice *algorithm achieves an approximation ratio of* $\frac{1}{2}$ *for* MaxProfit-PC *in rings.*

Furthermore, observe that the MaxProfit-PC instance illustrated in Figure 1 also serves as a tight example for the Best Choice algorithm.

Algorithm 3 Iterative

Input: an instance $\langle G, \mathcal{P}, w, k\rangle$ of MaxProfit-PC, where G is a ring

for each color i **do**
  $\mathcal{S}_i := \emptyset$
  for each path $p \in \mathcal{P}$ **do**
    $\mathcal{S}_p := \{p\}$
    Find a maximum-profit set of edge-disjoint paths that do not overlap with p by running the Carlisle-Lloyd Algorithm for MaxProfit-PC on the instance $\langle G - p, \mathcal{P}^{\neg p}, w, 1\rangle$, where $G - p$ is the graph obtained by removing all edges of path p from G and $\mathcal{P}^{\neg p}$ is the set of paths that do not overlap with path p; insert these paths in $\mathcal{S}_p$.
    if $w(\mathcal{S}_p) > w(\mathcal{S}_i)$ **then**
      $\mathcal{S}_i := \mathcal{S}_p$
    end if
  end for
  Color all requests in $\mathcal{S}_i$ with color i.
  $\mathcal{P} := \mathcal{P} \setminus \mathcal{S}_i$
end for

Time complexity of Best Choice*:* Step 2 requires $O(km \log m)$ time. The selection of the k most profitable paths in Step 3 can be done in $O(m)$ time, by selecting the path with the $(|\mathcal{P}_e| - k)$-th smallest profit using a known linear time selection algorithm which at the same time performs a partition with the selected element as pivot (see e.g. [8], pages 189-191). Therefore, the overall time complexity of the algorithm is dominated by Step 2 and is $O(km \log m)$.

4.2 Iterative

In this subsection we present an algorithm that iteratively colors a maximum-profit subset of non-overlapping requests. This algorithm is based on a known maximum coverage technique that also applies to coloring problems (see e.g. Wan and Liu [1], Erlebach et al. [9], or Awerbuch et al. [10]). We will refer to this algorithm as Iterative. Iterative works as follows: during each iteration i it computes for each path p a maximum-profit subset $\mathcal{S}_p$ of non-overlapping paths that contains p. Finally, the set $\mathcal{S}_p$ with maximum profit is colored with color i and is removed from $\mathcal{P}$.

In order to compute $\mathcal{S}_p$ it suffices to solve MaxProfit-PC in the following instance: $\langle G - p, \mathcal{P}^{\neg p}, w, 1\rangle$, where $G - p$ is the graph obtained by removing all edges of path p from G and $\mathcal{P}^{\neg p}$ is the set of paths that do not overlap with path p. Observe that this instance is a chain instance and can be solved optimally with the Carlisle-Lloyd Algorithm. The solution of this instance, together with path p, constitutes the set $\mathcal{S}_p$. We give a detailed description of the algorithm in Algorithm 3.

It has been observed by Erlebach et al. [9] that a straightforward adaptation of the technique of Awerbuch et al. [10] can be used to prove that the Iterative algorithm achieves an approximation ratio of $1 - \frac{1}{e}$ for the cardinality version

Algorithm 4 MPLU-Greedy

Input: an instance $\langle G, \mathcal{P}, w, k \rangle$ of MaxProfit-PC

1: Sort the paths $p \in \mathcal{P}$ in order of non-increasing ratio $\frac{w(p)}{length(p)}$.
2: **for each** path $p \in \mathcal{P}$ (in the order of Step 1) **do**
3: If there is some color i that can be assigned to p, color path p with color i.
4: **end for**

of MaxProfit-PC, where all requests have profit equal to 1. It turns out that the analysis goes through for the case of non-uniform profits as well. We include the proof in Appendix A for the sake of completeness.

Theorem 3. *The* Iterative *algorithm achieves an approximation ratio of* $1 - \frac{1}{e}$ *for* MaxProfit-PC *in rings.*

Time complexity of Iterative*:* The time complexity of Step 5 of the algorithm is $O(km \log m)$, and in the worst case at most km iterations of the inner loop are needed. Therefore, the total time complexity of Iterative is $O(k^2 m^2 \log m)$.

4.3 Greedy

We present a natural greedy heuristic for MaxProfit-PC. The key idea is that the more edges a path uses, the more likely it is to block other, possibly more profitable paths from being added to the solution. On the other hand, a path may be so profitable that it is worth picking it in the solution, despite its length. Translating these observations into an algorithm, we end up with the following approach: consider the paths in non-increasing order of the ratio of their profit over their length; if there is an available color for the current path, color it–otherwise drop this path. We call this algorithm Most Profit per Length Unit Greedy, for short MPLU-Greedy (Algorithm 4). It is very fast and easy to implement but, as we show below, there is no constant ρ, $0 < \rho \leq 1$, such that the profit of the solution returned by the algorithm is guaranteed to be at least a fraction ρ of the optimal. Note that the algorithm works in any network topology, not just in rings.

Example 2 (Non-constant approximation ratio of MPLU-Greedy *in rings).* Consider the instance of MaxProfit-PC that is illustrated in Figure 2. There is only one available color, and two overlapping paths, p_1 and p_2, with $w(p_1) = \ell - 1$ and $w(p_2) = 1$. The length of the paths p_1 and p_2 is ℓ and 1, respectively. The MPLU-Greedy algorithm will first consider path p_2 and color it with the only available color. This will result in the path p_1 remaining uncolored. The total profit of this solution is 1. On the other hand, the optimal solution would use the only available color to color path p_1 and obtain a profit of $\ell - 1$. This implies that the solution returned by the algorithm can be as bad as a fraction $\frac{1}{\ell - 1}$ of the optimal. Given that ℓ can be arbitrarily large, the algorithm can be made to perform arbitrarily badly.

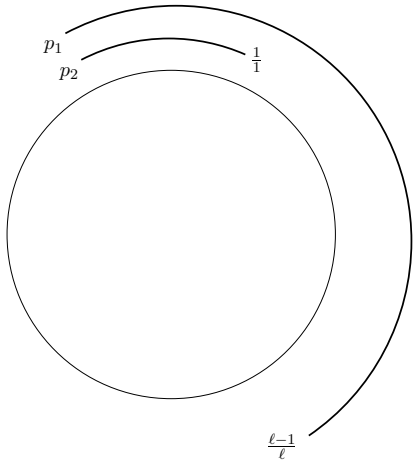

Fig. 2. An instance of MaxProfit-PC in which the MPLU-Greedy algorithm performs badly. There is only one available color and two paths, p_1 and p_2 with profits $\ell - 1$ and 1 respectively, and length ℓ and 1 respectively. The MPLU-Greedy algorithm will color path p_2 for a profit of 1, while the optimal solution would be to color path p_1 for a profit of $\ell - 1$. The value of ℓ is arbitrary.

Time complexity of MPLU-Greedy*:* A simple implementation of the algorithm requires $O(nmk)$ time.

5 Numerical Results

5.1 Experimental Setup

We implemented all algorithms in C++, making use of the LEDA$^{\text{TM}}$ class library of efficient data types and algorithms. All source files were compiled with the Borland$^{\text{TM}}$ C++ 5.5 for Win32 compiler, set to generate fastest possible code. We relied on LEDA routines and classes for graph, array, list and priority queue operations including sorting and finding maximum-weight matchings in bipartite graphs. The experiments were run on a Pentium$^{\text{TM}}$ 4 3.2GHz with 512MB of memory.

Instance packs: An *instance pack* is a set of 50 randomly generated MaxProfit-PC instances $\langle G, \mathcal{P}, w, k \rangle$, where G is a ring, specified by the following parameters:

- the number n of nodes in the ring,
- the number m of requests in the set $\mathcal{P}$,
- the number k of available colors,
- an upper bound W on the profit of the requests, and

– the manner in which paths are generated: either *uniform* or *gaussian:μ:σ*.

Each instance in the instance pack is defined on a ring with n nodes. There are k available colors. The path set $\mathcal{P}$ of the instance has cardinality m, and the profit of each path in $\mathcal{P}$ is selected uniformly at random from the set $\{1, \ldots, W\}$. The path itself is generated in one of two ways:

– If the mode of generation is *uniform*, the two endpoints of the path are selected independently uniformly at random from the node set of the ring. The edges actually used by the path are the edges that connect the first endpoint to the second one, in the clockwise direction.
– If the mode of generation is *gaussian:μ:σ*, then the first endpoint of the path is selected uniformly at random from the node set of the ring. Subsequently, the length of the path is selected at random, following the normal distribution with mean μ and standard deviation σ. The path spans as many edges as its length in the clockwise direction, starting from the first endpoint.

For each instance pack that we generated, we executed each algorithm on all instances of the pack and measured the average execution time and the average profit of satisfied requests. Furthermore, for each of these values we calculated a 95 percent confidence interval which is shown on the plots.

In each one of the figures discussed below (Figures 3, 4, 5, 6, 7), we present the results corresponding to several instance packs. In each of these instance packs, we keep four of the above parameters fixed and let one of them vary in order to exhibit the effect of this parameter on the execution time and on the profit of satisfied requests.

Note that execution times were measured using the *timer* class of the LEDA package, which does not provide routines for measuring exact processor time. However, we ran the experiments on a dedicated machine and kept background processes at a minimum.

Computing an upper bound on OPT: In order to obtain an estimation of the performance of our algorithms we use the following upper bound on the value of an optimal solution:

$$OPT \leq \min_{e \in E} \{ OPT_e + OPT_c \} \ ,$$

where OPT_e is the total profit of the k most profitable paths using edge e, and OPT_c is the optimal solution of the MaxProfit-PC instance that contains only the paths that *do not* use edge e. The latter is computed using the Carlisle-Lloyd Algorithm, as discussed earlier.

5.2 Discussion

A first observation is that all algorithms perform considerably better than their theoretical guarantee. Indeed, we have included a curve showing the computed upper bound (UB) in our figures and it turns out that all algorithms manage

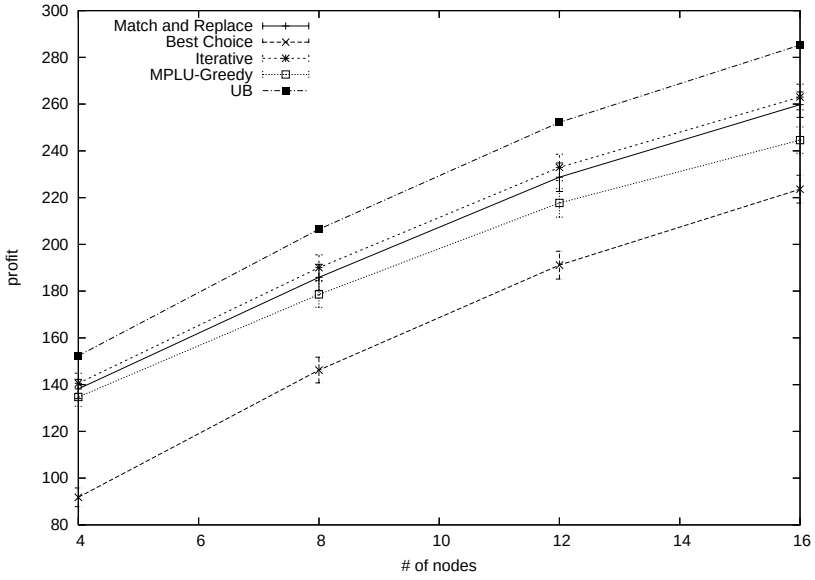

Fig. 3. Instance pack parameters: n ranges from 4 to 16, $m = 10n$, $k = 8$, $W = 10$, endpoints: *uniform*.

to satisfy a good fraction of an optimal solution, often much better than the theoretically predicted.

In the experiments of Figure 3, we compare all algorithms for variable number of nodes ranging from 4 to 16 (typical values for SONET rings), with the number of requests being ten times the number of nodes. The number of available wavelengths is fixed to 8. The endpoints of each request are chosen uniformly at random. The profit of each request is chosen uniformly at random from $\{1, \ldots, 10\}$. We observe that Iterative achieves the best performance, closely followed by Match-and-Replace which is a remarkably faster algorithm. MPLU-Greedy performs quite well, although there is no constant bound on its approximation ratio. Best Choice has no particular merits, but serves as a good basis in order to exhibit the improvement achieved by Match-and-Replace. In Figure 4, experiments with a wider variance of profits than the ones in Figure 3, namely between 1 and 100, result in a similar ranking of the algorithms in terms of achieved profit. Observe that, in some cases in Figure 4, Match-and-Replace even outperforms the Iterative algorithm (for example in the instance pack with $m = 350$).

In the experiments of Figure 5 the load is comparable to the number of wavelengths and this explains the good performance of all algorithms (except Best Choice). In particular, the average load is around 20 for every 100 requests and thus there are enough wavelengths to color almost all paths, especially for number of requests up to 300.

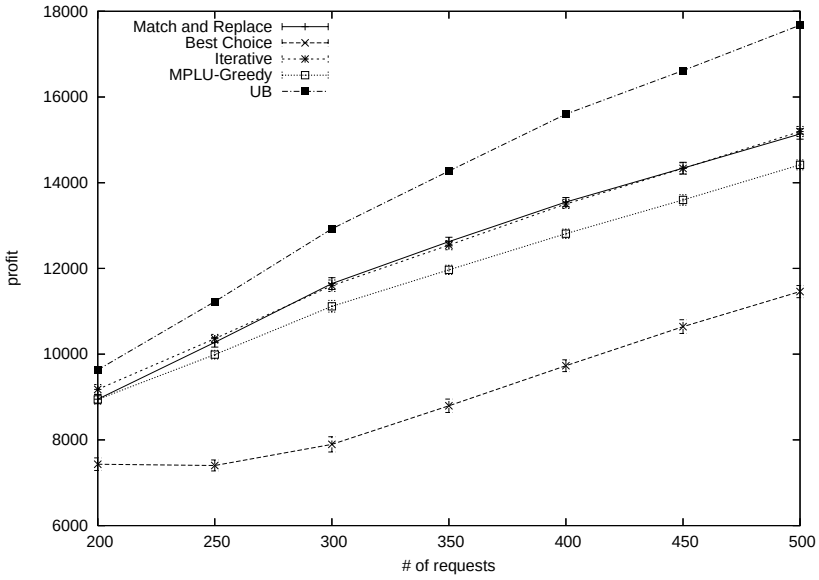

Fig. 4. Instance pack parameters: $n = 100$, m ranges from 200 to 500, $k = 80$, $W = 100$, endpoints: *uniform*.

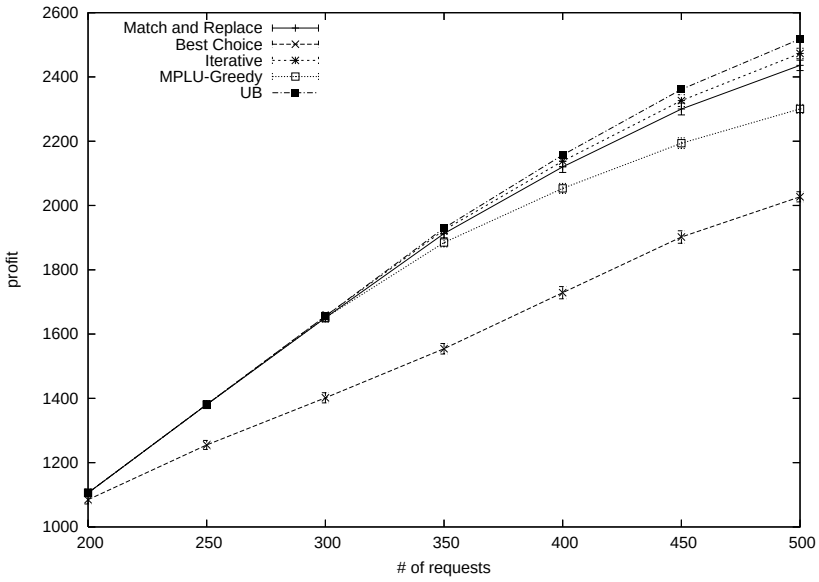

Fig. 5. Instance pack parameters: $n = 100$, m ranges from 200 to 500, $k = 80$, $W = 10$, endpoints: *gaussian:*20*:*2.

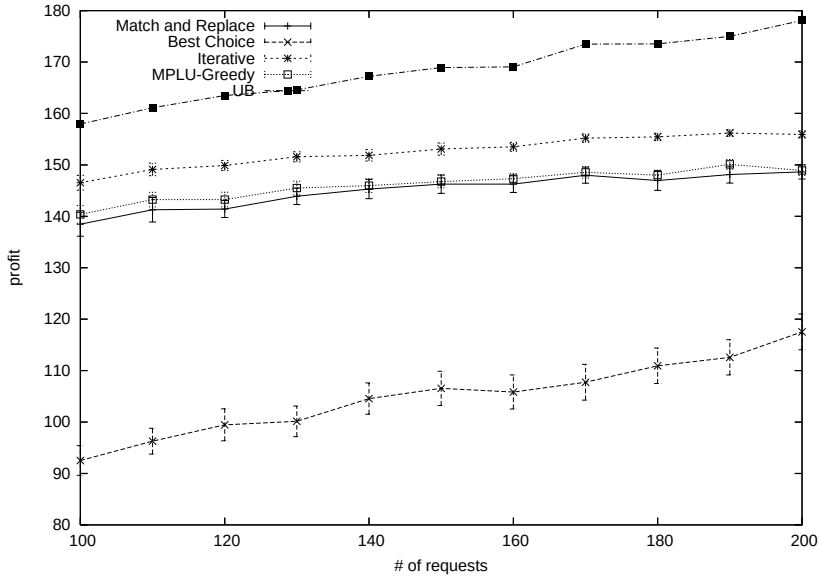

Fig. 6. Instance pack parameters: $n = 16$, m ranges from 100 to 200, $k = 8$, $W = 10$, endpoints: *gaussian:*8*:*1.

In the experiments of Figure 6 only one endpoint of each request is chosen uniformly at random and the other endpoint is determined in such a way so that the length of the request follows the normal distribution with mean 8 and standard deviation 1. In these experiments MPLU-Greedy appears competent and Iterative displays noticeable superiority. This behavior can be explained if we take into account that the length of paths is about half the cycle and thus with high probability each color can be used for at most two paths. This fact favors Iterative, which has fewer limitations on the path combinations it tries.

Figure 7 illustrates a comparison of the running time of the algorithms. We observe that Iterative is thousands of times slower than Match-and-Replace, which has comparable performance in terms of achieved profit. Best Choice is somewhat faster than Match-and-Replace. MPLU-Greedy is several times faster than Best Choice.

6 Conclusions

To evaluate the experimental results we take into consideration the obtained profit as well as the time performance. Taking into account both measures we first remark that Match-and-Replace should be the algorithm of choice for practical purposes, since it achieves one of the best performances with respect to the obtained profit, and at the same time its time requirements are reasonably low. In most cases Iterative produces marginally better solutions than Match-and-

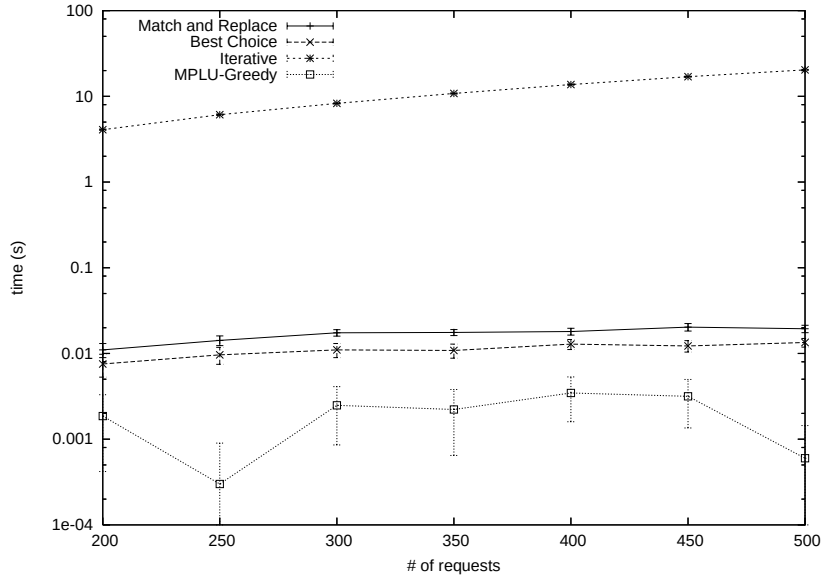

Fig. 7. Instance pack parameters: $n = 100$, m ranges from 200 to 500, $k = 80$, $W = 100$, endpoints: *uniform*.

Replace, but its time consumption could be prohibitive. On the other hand, if time efficiency is crucial it would also make sense to consider MPLU-Greedy, which is a very fast algorithm with acceptable performance. Taking into account both performance with respect to profit and time efficiency, as they were assessed from the experimental results, we rank all four algorithms in Table 1. We also include the theoretical bounds on the approximation ratio and time complexity for reference.

Table 1. A rough ranking of the algorithms with respect to their performance in the experiments. The two rightmost columns give the theoretical bounds on the approximation ratio and the time complexity.

Algorithm	Attained profit	Time efficiency	Appr. ratio	Time complexity
Match-and-Replace	★★★★	★★★	0.5	$O(m^2(k + \log m))$
MPLU-Greedy	★★★	★★★★	non-constant	$O(nmk)$
Iterative	★★★★	★	0.632	$O(k^2m^2 \log m)$
Best Choice	★	★★★	0.5	$O(km \log m)$

References

1. Wan, P.J., Liu, L.: Maximal throughput in wavelength-routed optical networks. In: Multichannel Optical Networks: Theory and Practice. Volume 46 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science., AMS (1998) 15–26
2. Erlebach, T., Jansen, K.: Maximizing the number of connections in optical tree networks. In Chwa, K.Y., Ibarra, O.H., eds.: ISAAC. Volume 1533 of Lecture Notes in Computer Science., Springer (1998) 179–188
3. Nomikos, C., Pagourtzis, A., Zachos, S.: Satisfying a maximum number of pre-routed requests in all-optical rings. Computer Networks **42**(1) (2003) 55–63
4. Caragiannis, I.: Wavelength management in WDM rings to maximize the number of connections. In Thomas, W., Weil, P., eds.: STACS. Volume 4393 of Lecture Notes in Computer Science., Springer (2007) 61–72
5. Carlisle, M.C., Lloyd, E.L.: On the k-coloring of intervals. Discrete Applied Mathematics **59**(3) (1995) 225–235
6. Bian, Z., Gu, Q.P.: 1.5-approximation algorithm for weighted maximum routing and wavelength assignment on rings. Inf. Process. Lett. **109** (2009) 400–404
7. Li, J., Li, K., Wang, L., Zhao, H.: Maximizing profits of routing in WDM networks. J. Comb. Optim. **10**(2) (2005) 99–111
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. First edn. McGraw Hill (1990)
9. Erlebach, T., Pagourtzis, A., Potika, K., Stefanakos, S.: Resource allocation problems in multifiber WDM tree networks. In Bodlaender, H.L., ed.: WG. Volume 2880 of Lecture Notes in Computer Science., Springer (2003) 218–229
10. Awerbuch, B., Azar, Y., Fiat, A., Leonardi, S., Rosén, A.: On-line competitive algorithms for call admission in optical networks. Algorithmica **31**(1) (2001) 29–43

A Proof of Theorem 3

Proof. Let $\langle G, \mathcal{P}, w, k\rangle$ be an input to the Iterative algorithm, and $t_i = w(\mathcal{S}_i)$, $1 \leq i \leq k$, be the total profit of the paths colored with color i during the i-th iteration of the algorithm. Let OPT be the total profit of an optimal solution.

We first prove that, for any $j : 1 \leq j \leq k$:

$$\sum_{i=1}^{j} t_i \geq OPT \cdot \left(1 - \left(1 - \frac{1}{k}\right)^j\right) \ . \tag{8}$$

Equation 8 certainly holds for $j = 1$: there is at least one set of non-overlapping edge-disjoint paths with total profit at least $\frac{OPT}{k}$, and the Iterative algorithm finds the largest such set during the first iteration. Assuming that Equation 8 holds for $j = s - 1$, we get:

$$\begin{aligned}
\sum_{i=1}^{s} t_i &= \sum_{i=1}^{s-1} t_i + t_s && (9)\\
&\geq \sum_{i=1}^{s-1} t_i + \frac{OPT - \sum_{i=1}^{s-1} t_i}{k} && (10)\\
&= \left(1 - \frac{1}{k}\right) \cdot \sum_{i=1}^{s-1} t_i + \frac{OPT}{k} && (11)\\
&\geq \left(1 - \frac{1}{k}\right) \cdot OPT \cdot \left(1 - \left(1 - \frac{1}{k}\right)^{s-1}\right) + \frac{OPT}{k} && (12)\\
&= OPT \cdot \left(1 - \left(1 - \frac{1}{k}\right)^{s}\right) \ . && (13)
\end{aligned}$$

Therefore, Equation 8 holds for all j between 1 and k. By setting $j = k$ in Equation 8 we get:

$$\sum_{i=1}^{k} t_i \geq OPT \cdot \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \geq \left(1 - \frac{1}{e}\right) \cdot OPT \ , \tag{14}$$

that is, the solution returned by the Iterative algorithm is at least a fraction of $1 - \frac{1}{e} \approx 0.632$ of the optimal. □