A 3/4-Approximation Algorithm for Maximum ATSP with Weights Zero and One

Markus Bläser

Institut für Theoretische Informatik, ETH Zürich
CH-8092 Zürich, Switzerland
mblaeser@inf.ethz.ch

Abstract. We present a polynomial time 3/4-approximation algorithm for the maximum asymmetric TSP with weights zero and one.
As applications, we get a 5/4-approximation algorithm for the (minimum) asymmetric TSP with weights one and two and a 3/4-approximation algorithm for the Maximum Directed Path Packing Problem.

1 Introduction

Traveling salesperson problems with weights one and two have been studied for many years. They are an important special case of traveling salesperson problems with triangle inequality. Papadimitriou and Yannakakis [8] showed that the undirected minimization problem is MaxSNP-complete. On the other hand, they presented a 7/6-approximation algorithm with polynomial running time. Vishwanathan [9] considered the corresponding asymmetric problem $\texttt{ATSP}(1,2)$ and gave a 17/12-approximation algorithm.

Let $\texttt{MaxATSP}(0,1)$ be the following problem: Given a directed complete loopless graph with edge weights zero and one, compute a TSP tour of *maximum* weight. $\texttt{MaxATSP}(0,1)$ is a generalization of $\texttt{ATSP}(1,2)$ in the following sense: Vishwanathan [9] showed that any $(1-\alpha)$-approximation algorithm for the former problem transforms into an $(1+\alpha)$-algorithm for the latter when replacing weight two with weight zero. (The other direction is not known to be true.)

By computing a matching of maximum weight and patching the edges together arbitrarily, one easily obtains a polynomial time 1/2-approximation algorithm for $\texttt{MaxATSP}(0,1)$. (Note that each edge has weight at least zero, thus we cannot loose any weight during the patching process.) Vishwanathan [9] was the first to improve on this by designing a 7/12-approximation algorithm with polynomial running time. In 1994, Kosaraju, Park, and Stein [6] gave a 48/63-approximation algorithm with polynomial time that also worked for maximum ATSP with arbitrary nonnegative weights. In their work, they also formulated the so-called path coloring lemma, which will be crucial for our algorithm. Bläser and Siebert [3] obtained a 4/3-approximation algorithm with running time $O(n^{5/2})$ for $\texttt{ATSP}(1,2)$. This algorithm can also be modified to give a 2/3-approximation algorithm for $\texttt{MaxATSP}(0,1)$ with the same running time [4]. Finally, Kaplan et al. [5] generalize this result by designing a 2/3-approximation

algorithm that works for maximum ATSP with arbitrary nonnegative weights but has a worse running time.

Closely related to MaxATSP$(0,1)$ is the Directed Path Packing Problem DPP. Here we are given a directed graph $G = (V, E)$. The aim is to find a subset P of node-disjoint paths of G such that the number of edges in P is maximized. By giving edges in G weight one and "non-edges" weight zero, any path packing transforms into a TSP tour by patching the paths arbitrarily together. On the other hand, any TSP tour yields a path packing by discarding all edges of weight zero. The only exception is the case where an optimum TSP tour has weight n. Here one weight one edge has to be discarded.

Our main result is a 3/4-approximation algorithm for MaxATSP$(0,1)$ with polynomial running time. As corollaries, we get a 5/4-approximation algorithm for ATSP$(1,2)$ and a 3/4-approximation algorithm for DPP.

1.1 Notations and Conventions

For a set of nodes V, $K(V)$ denotes the set of all edges (u, v) with $u \neq v$. In other words, $(V, K(V))$ is the complete loopless graph with node set V.

For a multigraph H and an edge e of H, $m_H(e) \in \mathbb{N}$ denotes the multiplicity of e in H, that is, the number of copies of e that are present in H. If H is clear from the context, we will also omit the subscript H.

A multigraph is called *2-path-colorable* if its edges can be colored with two colors such that each color class is a collection of node-disjoint paths. (Double edges are not allowed in such a collection of paths.)

1.2 Outline of our Algorithm

Kosaraju, Park, and Stein [6] formulate the so-called path coloring lemma. It states that if each node of a multigraph H has indegree and outdegree at most two and total degree at most three and H does not contain any 2-cycles (that is, a cycle with exactly two edges) or triple edges, then H is 2-path colorable. Kosaraju, Park, and Stein proceed with computing a cycle cover and a matching. (A cycle cover of a graph is a collection of node-disjoint directed cycles such that each node belongs to exactly one cycle.) If the matching is carefully chosen, then the union of the cycle cover and the matching fulfills the premises of the path coloring lemma and henceforth, is 2-path-colorable. (One also has to deal with the 2-cycles in the cycle cover separately, the interested reader is referred to the original work.) If one now takes the color class with the larger weight and patches the paths arbitrarily together, one gets a TSP tour that has at least half the weight of the combined weight of the cycle cover and the matching. The weight of an optimum cycle cover is at least the weight of an optimum TSP tour and the weight of an optimum matching is at least half the weight of an optimum TSP tour. Thus in the ideal case, this would yield an 3/4-approximation. However, Kosaraju, Park, and Stein have to deal with 2-cycles and have to avoid triple edges. Therefore, they only get a 48/63-approximation. This approach is refined in subsequent works [2, 7].

In this work, we directly compute a maximum weight multigraph that fulfills the premises of the path coloring lemma. This is done via an LP approach (Section 3). The fractional solution H^* is then rounded to an integer one via an iterated decomposition scheme (Section 4). Finally the integer solution is transformed into a TSP tour (or Path Packing) via the path coloring lemma.

2 Admissible Multigraphs

A directed multigraph is called *admissible*, if

1. the indegree and outdegree of each node is at most two,
2. the total degree of each node is at most three,
3. between each pair of nodes, there are at most two edges (counted with multiplicities).

Let $\omega : K(V) \to \{0, 1\}$ be a weight function. (This will be the weight function of the input of our algorithm for $\texttt{MaxATSP}(0, 1)$.) Our goal is to find an admissible multigraph H of maximum weight where the edges are weighted according to ω, that is, each edge contributes weight $m_H(e) \cdot \omega(e)$.

Lewenstein and Sviridenko [7], by reduction to the path coloring lemma of Kosaraju, Park, and Stein [6] (see Bläser [2] for a proof), show the following variant of the path coloring lemma:

Lemma 1 (Path coloring lemma). *If there are no 2-cycles on a cycle in an admissible multigraph G, then G is 2-path-colorable.*

Above, a 2-cycle on a cycle is a directed cycle $v_1, \ldots, v_k, v_1$ with $k \geq 3$ such that (v_i, v_{i-1}) (if $i = 1$, then $i - 1$ is k) is also an edge for some i.

We first compute an admissible multigraph of maximum weight. Then we have to remove 2-cycles on a cycle. In the case of weights zero and one, we are able to remove these 2-cycles without any loss of weight.

3 LP for Maximum Weight Admissible Multigraphs

For each edge $e \in K(V)$, there is a variable x_e. If $e = (u, v)$, we also write x_{uv} instead of x_e. The following integral LP solves the problem of finding a maximum weight admissible multigraph:

$$\text{Maximize } \sum_{e \in K(V)} \omega(e) x_e \text{ subject to}$$

$$\sum_{u \neq v} x_{uv} \leq 2 \qquad \text{for all } v \in V \qquad \text{(indegree)}$$

$$\sum_{v \neq u} x_{uv} \leq 2 \qquad \text{for all } u \in V \qquad \text{(outdegree)}$$

$$\sum_{v \neq u} x_{uv} + \sum_{w \neq u} x_{wu} \leq 3 \qquad \text{for all } u \in V \qquad \text{(total degree)}$$

$$x_{uv} + x_{vu} \leq 2 \qquad \text{for all } u, v \in V,\ u \neq v \qquad \text{(triple edge)}$$

$$x_{uv} \in \{0, 1, 2\} \qquad \text{for all } u, v \in V,\ u \neq v \qquad \text{(multiplicity)}$$

Constraints (outdegree) and (indegree) assure that the outdegree and indegree of each node are at most two. By (total degree), each node has total degree at most three. Constraint (triple edge) forbids more than two edges (counted with multiplicities) between each pair of nodes. The last condition ensures that each edge has integral multiplicity. By (triple edge), each edge can have multiplicity at most two. We now relax the integrality constraint:

$$0 \leq x_{uv} \leq 2 \quad \text{for all } u, v \in V,\ u \neq v \qquad (\text{multiplicity}')$$

(Note that only the lower bound of (multiplicity$'$) is actually needed, the upper bound also follows from (triple edge).) Let x^*_{uv} be an optimal solution of the relaxed LP. Let ω^* be its weight. We may assume that the x^*_{uv} are rational numbers. Their smallest common denominator D can be represented with poly(n) bits. We define the multigraph H^* as follows. For all u and v with $u \neq v$, there are $x_{uv} \cdot D$ copies of the edge (u, v) in H^*. Furthermore, we define the bipartite multigraph B^* with bipartition $V \cup V'$ (V' is a copy of V) as follows: If there are d copies of the edge (u, v) in H^*, then there are d copies of the edge (u, v') in B^*. The graph B^* will be helpful in the decomposition of H^*. By construction, H^* and B^* fulfill the following properties:

(P1) The outdegree of each node v in H^* is at most $2D$. The degree of each node $v \in V$ in B^* is at most $2D$.
(P2) The indegree of each node v in H^* is at most $2D$. The degree of each node $v' \in V'$ in B^* is at most $2D$.
(P3) The total degree of each node v in H^* is at most $3D$. The sum of the degrees of v and v' in B^* is at most $3D$.
(P4) Between each pair of distinct nodes u and v there are at most $2D$ edges in H^*. For any pair of distinct nodes u and v, the sum of the number of edges between u and v' and between v and u' is at most $2D$.

4 A Rounding Procedure for H^*

We first assume that $D = 2^\delta$ is a power of two. We will deal with the general case later. We now decompose H^* into two subgraphs H^*_1 and H^*_2 such that both H^*_1 and H^*_2 fulfill the properties (P1)–(P4) of the preceding section with D replaced by $D/2$. If we now replace H^* with the heavier of H^*_1 and H^*_2 and proceed recursively, we will end up with an optimum admissible multigraph after $\log D = \text{poly}(n)$ such decomposition steps.

In the following, H denotes the graph of the current iteration of the process outlined above and B is the corresponding bipartite graph. Our rounding procedure uses the fact that the edge weights are either zero or one.

Alon [1] gives a similar procedure for edge-coloring bipartite multigraphs. This procedure is then used by Kaplan et al. [5] for decomposing a fractional solution of an LP for computing cycle covers.

4.1 Normalization of H

We first ensure that H fulfills also the following property (P5) in addition to (P1)–(P4).

(P5) For all u and v: If $m(u,v) + m(v,u) = 2D$, then $m(u,v)$ and $m(v,u)$ are both even.

Algorithm 1 takes a multigraph H that fulfills (P1)–(P4) and transforms it into a multigraph H' with the same weight that fulfills (P1)–(P5).

The algorithm basically works as follows: Let u and v be two nodes. W.l.o.g. $m(u,v) \geq m(v,u)$. Furthermore, assume that $m(u,v) + m(v,u) = 2D$ and both $m(u,v)$ and $m(v,u)$ are odd. (As D is even, if one of $m(u,v)$ and $m(v,u)$ is odd, then both are.) Since D is a power of two, $m(u,v) > D$.

If $\omega(u,v) = 0$, then we can simply remove all copies of (u,v). This does not change the weight of H. Furthermore, this does not affect (P1)–(P4), too. If $\omega(v,u) = 0$, then we remove all copies of (v,u). The interesting case is $\omega(v,u) = \omega(u,v) = 1$. Here we remove one copy of (u,v) and add one copy of (v,u). This does not change the weight. Since $m(u,v) > D$, $m(u,v) \geq D$ thereafter. In particular, the indegree of u and the outdegree of v are both still at most $2D$. Therefore, the resulting graph H' fulfills (P1)–(P4), too.

By construction, H' fulfills (P5). This shows the following lemma.

Lemma 2. *Given a multigraph H fulfilling (P1)–(P4) for some $D = 2^\delta$, Algorithm 1 computes a multigraph H' that has the same weight as H and fulfills (P1)–(P5) for D.*

4.2 Decomposition of H

Next we describe how to decompose a graph H fulfilling (P1)–(P5) for D into two graphs H_1 and H_2 fulfilling (P1)–(P4) for $D/2$. Let H_1 be the heavier of H_1 and H_2. We thereafter normalize H_1 and proceed recursively.

In the first for loop of Algorithm 2, we run through all edges (u,v). If the multiplicity of (u,v) is even in H, then we simply divide all the copies evenly between H_1 and H_2. If the multiplicity is odd, then we divide all but one copy evenly between H_1 and H_2. Thereafter, all edges in H have multiplicity zero or one.

If the indegree and outdegree of a node u in H are both odd, then we add the edge (u,u') to B. Then we take two nodes of odd degree in a connected component of B. (If one such node exists, then there must exist another one.) We compute a path P connecting two such nodes and add the edges of P in an alternating way to H_1 and H_2. If there are not any nodes of odd degree, each connected component of B is Eulerian. We compute Eulerian tours and distribute the edges in the same way as in the case of a path P.

We claim that H_1 and H_2 fulfill (P1)–(P4) with parameter $D/2$. For (P1) note that we always remove the edges in pairs. If for instance an edge (u,v) is

Algorithm 1 Normalization of H

Input: Multigraph H fulfilling (P1)–(P4) for $D = 2^\delta$
Output: Multigraph H' fulfilling $\omega(H) = \omega(H')$ and (P1)–(P5) for D
  for all unordered pairs of nodes $u, v \in V$, $u \neq v$ **do**
    Let m be the multiplicity of (u, v) in H and m' be the multiplicity of (v, u) in H. W.l.o.g. $m \geq m'$.
    if $m + m' < 2D$ or m or m' is even **then**
      insert (u, v) and (v, u) with multiplicities m and m' into H', respectively.
    else
      if $\omega(u, v) = 0$ and $\omega(v, u) = 1$ **then**
        insert m' copies of (v, u) into H'.
      end if
      if $\omega(u, v) = 1$ and $\omega(v, u) = 0$ **then**
        insert m copies of (u, v) into H'.
      end if
      if $\omega(u, v) = \omega(v, u) = 1$ **then**
        insert $m - 1$ copies of (u, v) insert $m' + 1$ copies of (v, u) into H'
      end if
    end if
  end for

removed from H and added to H_1, then also an edge (u, w) is remove from H and added to H_2. In other words, the outdegree of u in H_1 and H_2 is always the same. The only exception is the case when the original outdegree of u is odd. Then one more edge is added to H_1, say, than to H_2. However, since the outdegree in H was odd, it was at most $D - 1$. Therefore, the degree in H_1 is at most D and in H_2, it is at most $D - 1$. The same argument works if the roles of H_1 and H_2 are interchanged. In the same way, we can show that H_1 and H_2 fulfill (P2) with $D/2$.

For (P3), we distinguish four cases, depending on the parity of the indegree i and outdegree o of a node u in H. If both i and u are even, then the indegree and outdegree of u are $i/2$ and $o/2$, respectively, in both H_1 and H_2 by construction. If both i and u are odd, then the indegree and outdegree are $\lfloor i/2 \rfloor$ and $\lceil o/2 \rceil$, respectively, in one of H_1 and H_2, and $\lceil i/2 \rceil$ and $\lfloor o/2 \rfloor$ in the other of H_1 and H_2. This is due to the fact that we added the edge (u, u') to B. In both cases, the indegree and outdegree sums up to $(i + o)/2 \leq 3D/2$. Finally, we consider the case where either i or o is odd. We assume that i is odd, the other case is treated symmetrically. Then the indegree of u in H_1, say, is $\lceil i/2 \rceil$ and in H_2, it is $\lfloor i/2 \rfloor$. In both subgraphs, the outdegree of u is $o/2$. The total degree of u in H is however at most $3D - 1$, since $3D$ is even. Therefore, $i + o \leq 3D - 1$ and $\lceil i/2 \rceil + o/2 \leq 3D/2$.

It remains to show that (P4) holds: Let m be the multiplicity of (u, v) in H and m' be the multiplicity of (v, u) in H. If both m and m' are even, then half of the copies of both (u, v) and (v, u) is added to H_1 and the other half are added to H_2 in the first for loop. Thus the number of edges between u and v is $(m + m')/2 \leq 2D/2$ in both H_1 and H_2. If m is odd and m' is even or vice

Algorithm 2 Decomposition of H

Input: Multigraph H fulfilling (P1)–(P5) for some $D = 2^\delta$
Output: Multigraphs H_1 and H_2 fulfilling (P1)–(P4) for $D/2$

```
for all (ordered) pairs of nodes u and v, u ≠ v do
  Let m be the multiplicity of (u, v).
  Let h = ⌊m/2⌋.
  Remove 2h copies of (u, v) from H.
  Add h copies to H_1 and h copies to H_2.
  Update B accordingly.
end for
for all nodes u do
  if the indegree and outdegree of u in H are both odd then
    add (u, u') to B
  end if
end for
while B contains nodes of odd degree do
  Choose two nodes a, b ∈ V ∪ V' of odd degree in B that lie in the same connected
  component.
  Compute a path P from a to b.
  Remove the edges of P from H and add them to H_1 and H_2 in an alternating way.
  Skip all edges of the form (u, u'). Update B.
end while
for each connected component C of B do
  Compute a Eulerian tour of C.
  Remove the edges of C from H and add them to H_1 and H_2 in an alternating way,
  again skipping all edges of the form (u, u').
end for
```

versa, then there are $\lfloor (m+m')/2 \rfloor$ between u and v in both H_1 and H_2 after the first for loop. Then one further copy is added to either H_1 and H_2. But since $2D$ is even, $\lfloor (m+m')/2 \rfloor < 2D/2$ and thus $\lfloor (m+m')/2 \rfloor + 1 \leq 2D/2$. The last case is the one where both m and m' are odd. Then $\lfloor m/2 \rfloor + \lfloor m'/2 \rfloor$ copies are added to H_1 and H_2, respectively, in the first for loop. In the remaining steps of Algorithm 2, two further copies are added to H_1 or H_2. In the worst case, they both go to one graph, say H_1. Since H fulfills (P5), $m + m' < 2D$. Thus $\lfloor m/2 \rfloor + \lfloor m'/2 \rfloor < D - 1$. Therefore, $\lfloor m/2 \rfloor + \lfloor m'/2 \rfloor + 2 \leq D = 2D/2$. Thus H_1 and H_2 also fulfill (P4) for $D/2$. Thus we obtain the next result.

Lemma 3. *Given a multigraph H that fulfills (P1)–(P5) for some $D = 2^\delta$, Algorithm 2 computes two multigraphs H_1 and H_2 such that both H_1 and H_2 fulfill (P1)–(P4) and for each edge e, $m_H(e) = m_{H_1}(e) + m_{H_2}(e)$.*

4.3 An Algorithm for Maximum Weight Admissible Multigraphs

Algorithm 3 repeatedly takes the multigraph H, normalizes it via Algorithm 1, decomposes it via Algorithm 2, and proceeds iteratively with the heavier of H_1 and H_2. Lemmas 2 and 3 immediately prove the following result.

Algorithm 3 Maximum weight admissible subgraph

Input: Multigraph H fulfilling (P1)–(P4) for some even $D = 2^\delta$
Output: Maximum weight admissible multigraph S
  for $i = 1, \ldots, \delta$ **do**
    Normalize H via Algorithm 1.
    Compute multigraphs H_1 and H_2 from H via Algorithm 2.
    Let w.lo.g. H_1 be the heavier of the two computed multigraphs.
    Set $H = H_1$.
  end for
  Set $S = H$.

Lemma 4. *Given a multigraph H that fulfills (P1)–(P4) for some $D = 2^\delta$ and $\omega(H) = \omega^*$, Algorithm 3 computes a maximum weight admissible multigraph S.*

4.4 Making D a Power of Two

If D is not a power of two, then we use the following standard trick and replace it by $\hat{D} = 2^{\hat{\delta}}$ where $\hat{\delta}$ is the smallest natural number such that $2n^2 D \leq 2^{\hat{\delta}}$. Each value of the optimal solution x^*_{uv} is rounded down to the next multiple of $2^{-\hat{\delta}}$. Let $\hat{x}_{uv}$ be the values obtained. We have

$$\hat{x}_{uv} \geq x^*_{uv} - 2^{-\hat{\delta}} \tag{1}$$

Let $\hat{H}$ be the multigraph that has $\hat{x}_{uv} \cdot \hat{D}$ copies of each edge (u, v).

Since we round each value down, $\hat{H}$ fulfills (P1)–(P4). It remains to estimate the loss of weight. By (1),

$$\begin{aligned} w(\hat{H}) &= \sum_{(u,v)\in K(V)} \omega(u,v)\hat{x}_{uv}2^{\hat{\delta}} \\ &\geq \omega^* 2^{\hat{\delta}} - \sum_{(u,v)\in K(V)} \omega(u,v)2^{-\hat{\delta}} \cdot 2^{\hat{\delta}} \\ &\geq \omega^* 2^{\hat{\delta}} - n^2, \end{aligned}$$

because $\omega(u, v) \in \{0, 1\}$. If we now run Algorithm 3 on the graph $\hat{H}$ for $\hat{\delta}$ iterations, then we end up with an admissible multigraph S of weight

$$\omega(S) \geq (\omega^* 2^{\hat{\delta}} - n^2)/2^{\hat{\delta}} \geq \omega^* - 1/(2D).$$

Therefore, $D\omega(S) \geq D\omega^* - 1/2$. Since both quantities $D\omega(S)$ and $D\omega^*$ are integers, we have that even $D\omega(S) \geq D\omega^*$.

Therefore, we have a polynomial time solution for computing maximum weight admissible multigraphs.

Theorem 1. *On input $\hat{H}$, Algorithm 3 computes a maximum weight admissible multigraph in polynomial time.*

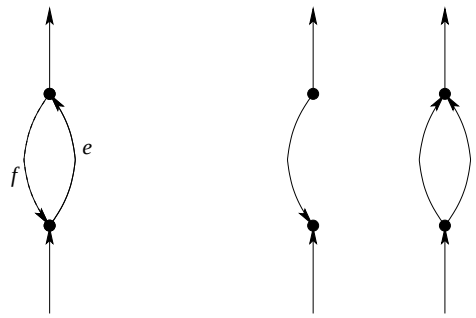

Fig. 1. On the lefthand side: A potential 2-cycle on a cycle. On the righthand side: The two ways how a potential 2-cycle on a cycle is treated.

5 Coloring Admissible Multigraphs

We show that for any admissible multigraph G, there is another admissible multigraph G' of the same weight that is even 2-path colorable. Given G, G' can be found in polynomial time. Our aim is to exploit the path coloring result by Lewenstein and Sviridenko. To do so, we have to deal with 2-cycles on a cycle.

A 2-cycle c on a cycle locally looks as depicted on the lefthand side of Figure 1. If e has weight zero, then we can simply discard e. This does not change the weight. The resulting graph is still admissible and c is no longer a 2-cycle on a cycle. If e has weight one, then we remove f and add another copy of e. This can only increase the weight, since the weight of f is either zero or one. The resulting graph is still admissible and c is no longer a 2-cycle on a cycle. (Note that the procedure of Lewenstein and Sviridenko can deal with double edges.)

We now consider all (unordered) pairs of nodes and deal with them as described above. This shows the following result.

Lemma 5. *Given an admissible multigraph G and a weight function $\omega : K(V) \to \{0, 1\}$, there is a 2-path-colorable admissible multigraph G' with $\omega(G) \leq \omega(G')$. Given G, G' can be found in polynomial time.*

6 Applications

Algorithm 4 now computes a maximum path packing or a maximum TSP tour, respectively. We first solve the fractional LP, round the optimum solution as in Section 4.2, make it 2-path-colorable, and finally take the color class of larger weight as a path packing, in the case of `DPP`, or patch the paths of this class together to form a TSP tour, in the case of `MaxATSP`$(0, 1)$.

Theorem 2. *Algorithm 4 is a polynomial time* $3/4$*-approximation algorithm for* `DPP` *and* `MaxATSP`$(0, 1)$.

Algorithm 4 DPP, MaxATSP$(0,1)$

Input: Directed Graph $G = (V, K(V))$ with weight function $\omega : K(V) \to \{0,1\}$
Output: A path packing or TSP tour, respectively

Solve the fractional LP in Section 3.
Round the optimum fractional solution H^* to an admissible multigraph S via Algorithm 3. (Replace H^* by $\hat{H}$, if necessary.)
Find a 2-path colorable admissible graph S' with the same weight as S.
Color the edges of S' with two colors such that each color class is a collection of node-disjoint paths.
Return the collection of larger weight. In the case of MaxATSP$(0,1)$, patch these path together arbitrarily.

Proof. By construction, the output of the algorithm is a feasible solution and it is computed in polynomial time. It remains to estimate the approximation performance.

If G has a path packing P with ℓ edges, then there is an admissible multigraph of weight $\frac{3}{2}\ell$. To see this, we decompose P into two matchings M_1 and M_2 by placing the edges of each path in P into M_1 and M_2 in an alternating fashion. Let w.lo.g. be M_1 the matching with more edges. Then $P \cup M_1$ is an admissible multigraph of weight $\frac{3}{2}\ell$.

In the same way we see that if there is a TSP tour of weight ℓ, there is an admissible multigraph of weight $\frac{3}{2}\ell$. (Strictly speaking, this is only true if the number of nodes is even. We can make it even by either adding a dummy node that is connected with weight zero edges to all other nodes. This gives a multiplicative loss of $(1-1/n)$ in the approximation factor. Or we can guess two consecutive edges and contract them into one. This increases the running time by a quadratic factor, but does not affect the approximation performance.)

The optimum admissible multigraph is divided into two color classes. Therefore, the heavier of the two classes has weight at least $\frac{3}{4}\ell$. □

Corollary 1. *There is a* $5/4$*-approximation algorithm with polynomial running time for* ATSP$(1,2)$.

Proof. Vishwanathan [9] showed that any $(1-\alpha)$-approximation algorithm for MaxATSP$(0,1)$ yields an $(1+\alpha)$-approximation for ATSP$(1,2)$, too. □

References

1. N. Alon. A simple algorithm for edge-coloring bipartite multigraphs. *Inform. Processing Letters* 85:301–302, 2003.
2. M. Bläser. An 8/13-approximation algorithm for the maximum asymmetric TSP. *J. Algorithms*, 50(1): 23–48, 2004.
3. M. Bläser and B. Siebert. Computing cycle covers without short cycles. In *Proc. 9th Ann. European Symp. on Algorithms (ESA)*, Lecture Notes in Comput. Sci. 2161, 369–379, Springer, 2001.

4. M. Bläser and B. Manthey. Approximating maximum weight cycle covers in directed graphs with edge weights zero and one. Manuscript, 2003.
5. H. Kaplan, M. Lewenstein, N. Shafrir, and M. Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. In *Proc. 44th Ann. IEEE Symp. on Foundations of Comput. Sci. (FOCS)*, pages 56–65, 2003.
6. S. R. Kosaraju, J. K. Park, and C. Stein. Long tours and short superstrings. In *Proc. 35th Ann. IEEE Symp. on Foundations of Comput. Sci. (FOCS)*, pages 166-177, 1994.
7. M. Lewenstein and M. Sviridenko. A 5/8 approximation algorithm for the maximum asymmetric TSP. *SIAM J. Discrete Math.*, 17:237–248, 2003.
8. C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Math. Oper. Res.*, 18:1–11, 1993.
9. S. Vishwanathan. An approximation algorithm for the asymmetric travelling salesman problem with distances one and two. *Inform. Proc. Letters*, 44:297–302, 1992.