The complexity of the bigamist matching problem

Klaus Reinhardt
Universität Tübingen, Germany,
reinhard@informatik.uni-tuebingen.de

April 12, 2006

Abstract

We describe a polynomial time algorithm to construct a maximal bigamist matching for a given bipartite graph with red and blue edges, that is the maximal set of vertex disjoint triples consisting of one bigamist vertex connected to two monogamist vertices with two different colors. This solves an open problem in [SGYB05]. As a method we use a "double reachability" algorithm to find a simple path in an undirected switch graph, which is equivalent to finding an augmenting path for the bigamist matchings. Furthermore, we show that some other problems of this kind are NP-complete.

1 Introduction

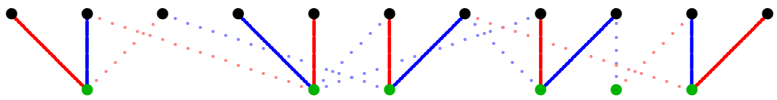
There is a well known polynomial time algorithm for constructing a maximal matching, that is a maximal set of vertex disjoint edges in a (possibly bipartite) graph. The idea of the algorithm is to continuously improve a matching as long as this is possible by searching for an augmenting path starting and ending with a previously unmatched vertex and consisting of alternating old and new edges. In [HK73] this is described as an $n^{5/2}$ algorithm.
In [ARZ99] it is shown that matching is in nonuniform SPL. On the other hand 3-dimensional matching (marriage for 3 sexes) is NP-complete [GJ78]. In more general terms, it is shown in [KH83] that it is NP-complete for any connected graph G with more than 2 vertices (like the "triangle" $G = K_3$ in 3-dimensional matching) to decide whether a given graph H can be "G-factored" (that means disjoint copies of G form a spanning subgraph of H). In particular, this holds for $G = P_3$ the path of length 2. Here, however, the input does not distinguish the vertices of the graph according to their

possible positions in G, i.e. which vertices can be the middle vertex of P_3. This will change in the following problem.

The maximum bigamist matching problem was introduced in [SGYB05] under the name Maximum Synchronized Matching problem and the authors described a polynomial-time approximation scheme.

Given: a bipartite graph $G = (B, M, E)$, where B is the set of bigamist vertices and M are the monogamists; additionally, the edges have the colors red and blue; thus, $E = E_r \cup E_b \subset B \times M$. The set of edges E is reflecting the "who would marry whom" relation.

Wanted: A maximum number of bigamists that could marry a monogamist over a red edge and, at the same time, another monogamist over a blue edge (otherwise a bigamist stays unmarried). In other words, a maximum number of pairwise vertex-disjoint triples (v, u, u') with $(v, u) \in E_r$ and $(v, u') \in E_b$.



The problem motivated by multiplexing schemes for generic SNP genotyping assays in [SGYB05] where the bigamists are allele-pairs (s,s') and the monogamists are features which are informative for s (blue edge) or s' (red edge). This suggests that the union $E = E_r \cup E_b$ is disjoint, for our solution however, we do not need this requirement and generalize in this way also over the case without distinction of the colors. This case without distinction of the colors could also be solved by repeatedly applying the polynomial time algorithm in [Cor88] for deciding Lovasz's general graph factoring problem[1] if there are no gaps > 1.

Note that the decision problem for a perfect bigamist matching is easily seen to be equivalent to a usual perfect matching by splitting the bigamists into red and blue "halfs". Therefore, the optimization problem is of particular interest.

In Section 2, we show that the method of iterating a solution by searching for an augmenting path can be transfered. But to search for such a path corresponds to a new kind of reachability problem. We formulate this as the reachability by a simple path in an undirected switch graph. In Section

[1] Given: A graph (V, E) and sets $B_v \subset \mathbb{N}$ for every $v \in V$. Question: Is there a subgraph (V, F) such that every vertex $v \in V$ has a degree in B_v? B_v has a gap > 1 if there exist $k, k + p \in B_v$ with $p > 2$ and $k + 1, k + 2 \notin B_v$. A bigamist v would get $B_v = \{0, 2\}$ and a monogamist u would get $B_u = \{0, 1\}$ (which has the trivial solution $F = \emptyset$) and a maximization algorithm would try to set $B_u = \{1\}$ for an increasing number of monogamists u.

3, we describe the double reachability algorithm to find a simple path in an undirected switch graph in $O(|v|^2)$ time. In Section 4, we consider corresponding problems which are NP-complete (here we consider the decision problems) in particular we show it for tetragamy.
This underlines the noteworthiness of the following main result which is obtained by starting with a possibly non-maximal bigamist matching and iteratively improving it according to Lemma 1 in at most $|M|/2$ many steps where Theorems 2 and 3 allow to find an augmenting path as long as the bigamist matching is not maximal:

Theorem 1 *A maximal bigamist matching for a given bipartite graph can be constructed in $O((|B| + |M|)^5)$ time.*

2 Finding an augmenting path as a simple undirected switch graph reachability problem

Like for the usual matching algorithms, we make use of the property that the symmetric difference of a bigamist matching and an improved bigamist matching must consist of paths (and cycles, which make no difference). At least one of them starts and ends with monogamists which are only married in the improved matching and on the way there are bigamists and monogamists which just change marriage and bigamist which were not married before or loose both partners.

Lemma 1 *If the bigamist matching is not maximal, we can always find such an augmenting path.*

Proof: If there is an improved bigamist matching, it contains more monogamists. Thus, we start constructing this path with a monogamist which is only married in the improved matching, which is (1) in the following cases:
Case (1): The bigamist which is now married to this monogamist was either married before, in this case we continue with (2) the previously married monogamist over the same edge color or this bigamist was not married before, in this case we continue with (3) the other monogamist which is now married over the other edge color.
Case (2): If the monogamist is not married anymore, there must (by a counting argument) be another monogamist which is only married in the improved matching and we start a new path from there; otherwise we continue with (1).
Case (3): If the monogamist was not married before, the augmenting path is

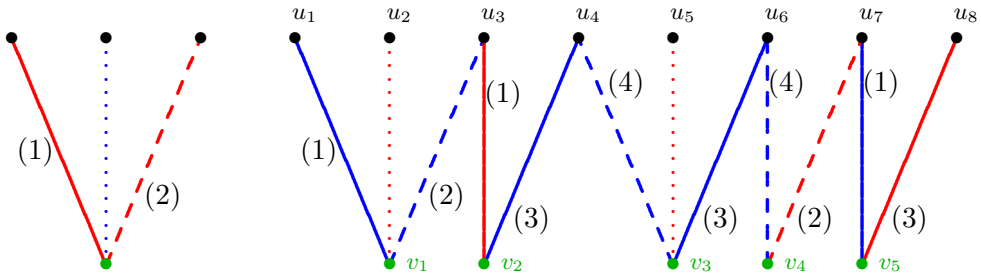


Figure 1: Example for an augmenting path where the previous bigamist matching consists of the dotted and dashed edges and the new matching consists of the dotted and solid edges. The numbers correspond to the construction case in the proof of Lemma 1.

complete; otherwise, we continue with (4) the bigamist previously married to this monogamist.
Case (4): Either this bigamist is still married and we continue with (3) the other monogamist which is now married over the same edge color or this bigamist is not married anymore and we continue with (2) the previously married monogamist over the other edge color.
Since the path can visit bigamist changing partners at most twice and monogamists at most once, it must terminate. ■
Now we will reduce the problem of finding an augmenting path for a given bigamist matching to finding a simple path in an undirected switch graph (such graphs appeared also in [Coo03]).

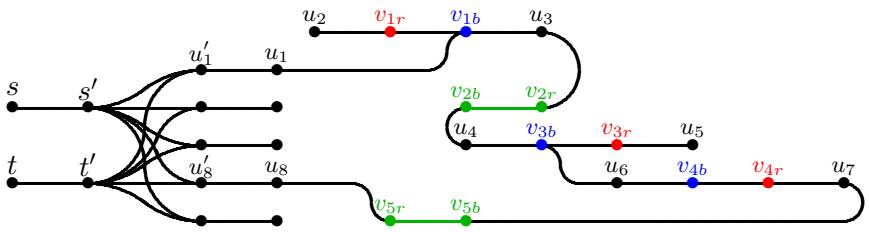
Definition 1 *An undirected* switch graph (V, E, o) *consists of "switches"* V*, undirected "tracks"* $E \subset V \times V$ *and a function* $o : V \mapsto E$ *which assigns an obligatory incident edge to each switch which means that a path going through a switch* v *has to use the "track"* $o(v) = (v, v')$ *either entering or exiting* v*. To make this more formal, we will write a path as a concatenation* $p = ...(u, v)(v, w)... \in E^*$ *of edges with the obvious condition that the second vertex of an edge is the first vertex of the following edge. We say that* p *contains an edge* $(u, v)^{rev} := (v, u)$ *in opposite direction and* $p^{rev} = ...(w, v)(v, u)....$ *Thus, the condition for* v *saying that* p *has to use* $o(v)$ *means more precisely that* $o(v) \in \{(v, u), (v, w)\}$ *in this example. Let* $|p|$ *denote the number of edges of a path. In the graphical representation, we might think of a path as the way a train could take*[2]*.*

[2]We may regard the setting of the switches as existentially quantified

The simple switch graph reachability *problem is the following: Given* (V, E, o) *and* $s, t \in V$*, is there a simple path* p *form* s *to* t*? This means* p *fulfills the above condition and each switch appears at most once in* p*.*

Theorem 2 *The problem of finding an augmenting path for a bigamist matching is logspace and linear time reducible to the simple undirected switch graph reachability problem.*

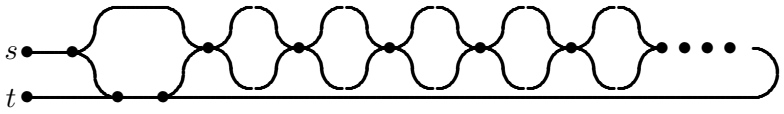
Proof: Given a possibly non-maximal bigamist matching on the bipartite graph $G = (B, M, E_r \cup E_b)$. The constructed switch graph (V, E, o) contains new vertices $s, s', t, t' \in V$ with $o(s) = o(s') = (s, s') \in E$ and $o(t) = o(t') = (t, t') \in E$. Each unmarried monogamist $u \in B$ is represented by two switches $u, u' \in V$ connected by the obligatory edge $o(u) = o(u') = (u, u') \in E$ and $(s', u'), (t', u') \in E$. Each unmarried bigamist $v \in B$ is represented by two switches $v_r, v_b \in V$ connected by the obligatory edge $o(v_r) = o(v_b) = (v_r, v_b) \in E$ (see left side of the previous picture). Each married bigamist $v \in B$ is represented by two switches $v_r, v_b \in V$ connected by the edge $(v_r, v_b) \in E$ and the monogamists $u_1, u_2 \in M$ married to v over a red respectively blue edge are connected by the obligatory edges $o(v_r) = o(u_1) = (v_r, u_1) \in E$ and $o(v_b) = o(u_2) = (u_2, v_b) \in E$ (see right side of the previous picture). Furthermore for each $(v, u) \in E_r$ we have the edge $(v_r, u) \in E$ and for each $(v, u) \in E_b$ we have the edge $(v_b, u) \in E$.
Find the corresponding path to the example in Figure 2 here:



Now it is easy to see that an augmenting path directly corresponds to a simple path in the switch graph. Note that it can happen that a bigamist vertex occurs twice in the augmenting path. In this case, it changes both partners in the improved matching; the corresponding simple path in the switch graph visits both obligatory edges of this bigamist independently. ■

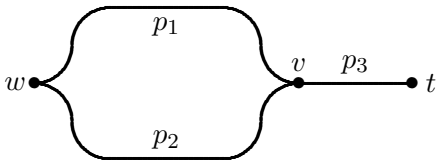
3 The double reachability method

Using an algorithm which marks switches reachable from s (like Dijkstra's algorithm) is not sufficient because, as the following example shows, we have to avoid using the same edge again in opposite direction.
A backtracking algorithm marking the path would require exponential time in the following situation:

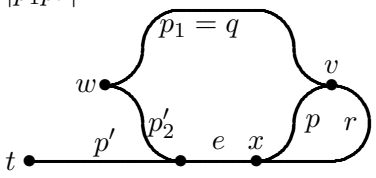


The solution of this problem is based on the following observation:

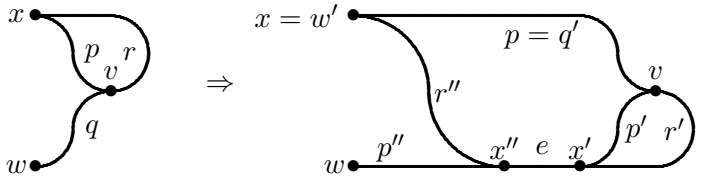
Lemma 2 *If we have two alternative simple paths p_1 and p_2 from w to u and a simple path p_3 containing $o(u)$ from u to t where p_1 and p_2 do not use the same edge in the same direction, then we can construct a simple path from w to t. Furthermore, the construction takes $\leq |p_1p_2p_3|^2$ steps.*



Proof: If p_i is disjoint with p_3 for some $i \leq 2$, then we can simply concatenate p_ip_3 to a simple path from w to t. Let $e = (x, x')$ be the last edge in p_3 which also occurs in p_i for some $i \leq 2$, that means $p_3 = rep'$ and p' is disjoint with p_1 and p_2. If e is used in both paths in the same direction, that means $p_i = p_i'ep$, then we can concatenate $p_i'ep'$ to a simple path from w to t. Otherwise, let w.l.o.g. e occur in p_2 in opposite direction (as the following picture shows), this means $p_2 = p_2'e^{rev}p$. Since p' is disjoint with p_1 and p_2, we can apply Lemma 3 with $q = p_1$ to construct a simple path from w to x and append ep'.
Finding e requires $|p'|$ times searching through p_1p_2. Together with $\leq |p_1pr|^2$ from Lemma 3, this is dominated by the time $|p_1p_2p_3|^2 \geq |p_1p_2p'|^2 + |p_1p_2r|^2 \geq |p_1p_2| \cdot |p'| + |p_1pr|^2$. ■



Lemma 3 *If we have a simple path r starting with $o(v)$ from v to x, a simple path p from x to v and a simple path q from w to v, where q and p do not use the same edge in the same direction, then we can construct a simple path from w to x. Furthermore, the construction takes $O(|prq|^2)$ time.*



Proof: If q and r are disjoint, then qr is a simple path from w to x. Let $e = (x'', x')$ be the first edge in q which also occurs in $p = p_1 e p_2$ or in $r = r' e r''$. In the first case, we must have $q = q_1 e^{rev} q_2$ (not the same direction) and $q_1 e^{rev} p_1^{rev}$ is a simple path from w to x. If, in the second case, e is used in q and r in the same direction, that means $q = q_1 e q_2$, then $q_1 e r''$ is a simple path from w to x. In the remaining case (right side of the above picture) we have $q = p'' e^{rev} p'$ and, since p'' is disjoint with p and r, we can apply Lemma 3 by induction over the length of r to construct a simple path from $w' = x$ to x' and append it reversed to $p'' e^{rev}$.
Finding e requires $|p''| \cdot |pr|$ searching steps. Together with $\leq |pr'p'|^2$ from the recursion, this is dominated by the time $|prq|^2 \geq |prp''|^2 + |prp'|^2 \geq |pr| \cdot |p''| + |pr'p'|^2$. ■

Lemma 2 justifies the introduction of a "shortcut edge" from w to v when two alternative paths p_1 and p_2 are found. This is used in the following algorithm.

Theorem 3 *The simple undirected switch graph reachability problem can be solved in polynomial time.*

Proof: The idea of the algorithm is a depth-first search in the graph, where the current path is remembered and switches which were visited from the direction of a non-obligatory edge keep the information from where they were visited. If the search enters a switch of the current path, it has to backtrack. If the search enters a switch visited before from the direction of a non-obligatory edge, the closest common ancestor w in the search tree gets a "shortcut edge" together with both path-informations to this switch and the search backtracks. The search always tries the newest unused "shortcut edge" from a switch.
To show the termination of this function, we prove the following:

```
Function Visit(path):
    Let (x, v) be the last edge in path;
    If v = t then return(path)
    else if o(v) = (v, x) then
        foundpath :=nil;
        Mark all incident non-obligatory edges "unvisited" for v;
        While ∃ "unvisited" edges for v and foundpath =nil do
            Let (v, u) be the newest unvisited edge;
            if u ∉ path then foundpath :=Visit(path ∘ (v, u));
            mark (v, u) "visited" for v;
        return(foundpath)
    else
        Let (w', w) = o(w) be the last edge in path occuring
            also in pathto(v) in the same direction
        if w = x or pathto(v)=nil then
            pathto(v) := path;
            return(Visit(path ∘ o(v)));
        else if (w, v) ∉pathto(v) then
            add edge (w, v); mark (w, v) "unvisited" for w;
            remember both paths from w to v for (w, u);
            return(nil)
        else return(nil)


Function expand(path):
    While ∃ a last shortcut edge (w, v) in p = q(w, v)p_3 do
        Let p_1, p_2 be the remembered paths from w to v;
        Construct a simple path r from w to t by Lemma 2;
        path := q ∘ r;
    return(path)
```

Figure 2: The pseudo code for the functions searching and expanding a path to t. Concatenating paths is described here by "$\circ$".

Lemma 4 *Each time the search algorithm in Figure 3 visits a vertex u using an obligatory edge, that means with the path $p(o(u))^{rev} = p(v,u)$, the path p has to be shorter than at the last such visit of u.*

Proof: One of the following two cases must hold:
(i) The edge $o(v) = (v,u)$ was also obligatory for v. This implies either pathto(u)=nil, which means that u is visited the first time, or the edge $(w',w) = o(w)$ for the vertex w of the edge (w,v) at the end of path p =pathto(w')(w',w) already occured in the previous visit of v with the path pathto$(v)(v,u)$. Then pathto$(w')(w',w)(w,v)(v,u)$ is shorter than pathto$(v)(v,u) = p'(w',w)p''(v,u)$ because the algorithm either backtracked on w, which means pathto$(w') = p'$ and p'' is unequal and thus longer than (w,v), or $p'' = (w,v)$ and p' is longer than pathto(w') by induction on the length of the path.
(ii) The path p' with $p = p'(o(v))^{rev}$ has to be shorter than at the last such visit of v by induction on the length of the path. ■
From Lemma 4 it follows that the algorithm will visit every switch at most $|V|$ times using an obligatory edge and thus, at most $|V|^2$ times using an obligatory edge. Finding w can be done in $\leq |V|^2$ steps. Going through all unvisited edges v and u and checking if u is not in *path* can also be done in $\leq |V|^2$ steps. Thus, the function Visit will terminate in $\leq |V|^4$ steps. When a path to t is found, it may contain shortcut edges which have to be expanded by Lemma 2 using the function expand.
All edges in p_1, p_2 and p_3 where visited with a path going through w and therefore p_1, p_2, p_3 and r must be disjoint with the path q in function expand. Since p_1 and p_2 contain only older shortcut edges than (w,v), each shortcut edge is expanded at most once in function expand. Thus, at most $|V|^2$ shortcut edges are each expanded in $\leq |V|^2$ steps by Lemma 2. This results in $\leq |V|^4$ steps to calculate expand and thus, $O(|V|^4)$ time for calculating the path expand(Visit$(o(s))$). ■
Together with Lemma 1 and Theorem 2 this completes the proof for Theorem 1.
Note here in contrast that the simple *directed* switch graph reachability problem can be shown to be NP-complete by a reduction from SAT: Each occurance of a variable is represented by an undirected edge which is obligatory for both of its points. The directed edges make sure that the path uses one such undirected edge for each clause in one direction and each such undirected edge for false variables in the other direction.

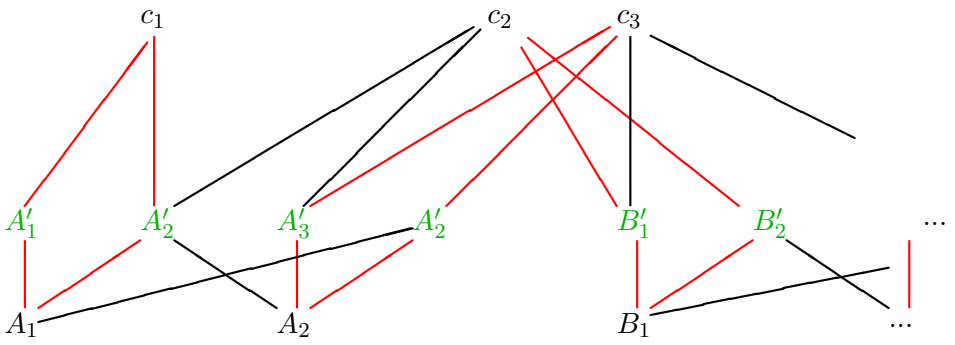


Figure 3: Example for a part of a tetragamist-input graph for $\phi = ... \wedge c_1 \wedge c_2 \wedge c_3 ...$ with $c_1 = (... \vee A ...), c_2 = (... \vee \overline{A} \vee B ...)$ and $c_3 = (... \vee A \vee \overline{B} ...)$ (one $\overline{A}$ occurs before c_1 and B's may occur after c_3.)

4 NP-complete cases

In this hardness-section we only need to consider the special case without the colors and the decision problem. It was shown in [Lov72] by reduction of planar 3-colorability that the general factor problem is NP-complete if sets $B_v = (0, 3)$ can occur. Similarly the trigamist matching problem[3] is NP-complete.
The philosopher Schopenhauer proposed the tetragamy in which two females and two males marry. The tetragamist matching problem analogously formulates as follows:
Given: A bipartite graph $G = (F, M, E)$ with edges $E \subset F \times M$.
Question: Are there $|F|/2$ pairwise vertex-disjoint quadruples (v, v', u, u') with $(v, u), (v, u'), (v', u), (v', u') \in E$?
With almost the same proof as in [GJ78], we can show the following:

Theorem 4 *The tetragamist matching problem is NP-complete*

Proof: Given a formula ϕ which we regard as set of clauses by abuse of notation, we construct a graph as follows (see Figure 4 for an example):
For each clause $c \in \phi$ we have a vertex in M. For each variable $A \in V_\phi$ (the set of variables occurring in ϕ) let n_A be the maximum of the number of occurrences of A in the formula and the number of occurrences of $\overline{A}$ in the formula and we use n_A copies A_i in M and $2n_A$ copies A'_i in F. Let $M' = \phi \cup \{A_1, ...A_{n_A} \mid A \in V_\phi\}$, $F = \{A'_1, ...A'_{2n_A} \mid A \in V_\phi\}$ and $E' =$

[3] here we look for vertex-disjoint quadruples (v, u, u', u'') with $(v, u), (v, u'), (v, u'') \in E$

$\{(A_i, A'_{2i}), (A_i, A'_{2i-1}), (A_i, A_{2(i-2 \bmod n_A)+2}{}')\} \cup \{(A'_{2i-1}, c), (A'_{2(i-2 \bmod n_A)+2}, c) \mid$ the i-th occurrence of A is in $c\} \cup \{(A'_{2i}, c), (A'_{2i-1}, c) \mid$ the i-th occurrence of $\overline{A}$ is in $c\}$. Since some clauses may contain more than one true variable in a satisfying assignment and some vertices A'_i might not have a connection to a clause, we have to fill up by $M = M' \cup \{m_i \mid 0 < i \leq |F| - |M'|\}$ and $E = E' \cup F \times \{m_i \mid 0 < i \leq |F| - |M'|\}$.
Because $|F| = |M|$ the matching must be perfect and by a counting argument every A'_i must be matched with some A_j. Thus, for every A a tetragamist matching either contains exactly the edges $(A_i, A'_{2i-1}), (A_i, A'_{2i})$ or exactly the edges $(A_i, A'_{2i-1}), (A_i, A'_{2(i-2 \bmod n_A)+2})$ which corresponds to A being true or false in the satisfying assignment. (All clauses have to be in the matching.) This establishes a one to one correspondence of tetragamist matchings and satisfying assignments for ϕ. ■
The main difference of the NP-complete problem from [KH83] of "G-factoring" graphs to the bigamist matching problem is, that the input does not specify which vertices of the graph can be covered by which vertex in G. This means that in the example of $G = P_3$, any vertex could be the middle vertex in the path (that means the bigamist). Here, NP-completeness also holds in the restriction to bipartite graphs. This can be seen easily with almost the same proof. In other words: The problem is NP-complete if we allow triples (v, u, u') with $(v, u) \in E$ and $(v, u') \in E$ together with triples (v, v', u') with $(v, u) \in E$ and $(v', u) \in E$.

5 Further possible work

We expect that the running time can be reduced from $O((|B| + |M|)^5)$ to a polynomial of lower degree by accompanying paths with more efficient data structures.
It seems that the complexity of the problem does not depend on the bipartiteness of the input: The algorithm can be modified or extended to replace the bigamist-triples by chains starting and ending with a monogamist and two bigamists in the middle or also allowing matchings of two monogamists or even allowing (the graph-family of) chains starting and ending with a monogamist and an arbitrary number of bigamists in the middle. So we may ask the following questions:
Is it possible to obtain a general characterization of (families of) graphs G together with such specifications which have the property that "G-factoring" the input graph can be done in polynomial time? Which conditions for more than 2 colors can be used?

What is the complexity of constructing a stable bigamist matching?
What is the complexity of constructing an optimum or a Nash-equilibrium for a bigamist matching with weighted edges?
Can the method in [ARZ99] for showing that matching is in nonuniform SPL be combined with the method of this paper? Is the bigamist matching problem in nonuniform FSPL or any other complexity class within P?

Acknowledgments

I thank Bernd Borchert, Henning Fernau, Jens Gramm and an anonymous referee for helpful conversations and comments.

References

[ARZ99] E. Allender, K. Reinhardt, and S. Zhou. Isolation matching and counting uniform amd nonuniform upper bounds. *Journal of Computer and System Sciences*, 59:164–181, 1999.

[Coo03] M. Cook. Still life theory. In Cristopher Moore David Griffeath, editor, *New Constructions in Cellular Automata*, volume 226, pages 93–118, Oxford University Press US, Mar 2003. Santa Fe Institute Studies on the Sciences of Complexity.

[Cor88] G. Cornuejols. General factors of graphs. *Journal fo Combinatorial Theory B*, 45:185–198, 1988.

[GJ78] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, San Francisco, 1978.

[HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[KH83] D.G. Kirkpatrik and P. Hell. On the complexity of general graph factor problems. *SIAM J. Comput.*, 12(3):601–608, 1983.

[Lov72] L. Lovasz. The factorization of graphs, II. *Acta Math. Acad. Sci. Hungar.*, 23:223–246, 1972.

[SGYB05] R. Sharan, J. Gramm, Z. Yakhini, and A. Ben-Dor. Multiplexing schemes for generic SNP genotyping assays. *Journal of Computational Biology*, 12:514–533, 2005.