Evaluation of ILP-based Approaches for Partitioning into Colorful Components

Sharon Bruckner[1]*, Falk Hüffner[2]**, Christian Komusiewicz[2], and Rolf Niedermeier[2]

[1] Institut für Mathematik, Freie Universität Berlin, Germany
sharonb@mi.fu-berlin.de
[2] Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
{falk.hueffner,christian.komusiewicz,rolf.niedermeier}@tu-berlin.de

Abstract. The NP-hard Colorful Components problem is a graph partitioning problem on vertex-colored graphs. We identify a new application of Colorful Components in the correction of Wikipedia interlanguage links, and describe and compare several exact and heuristic approaches. In particular, we devise two ILP formulations, one based on a Hitting Set and the one based on a Clique Partition formulation. Furthermore, we use the recently proposed implicit hitting set framework [Karp, JCSS 2011, Chandrasekaran et al., SODA 2011] to solve Colorful Components. Finally, we study a move-based and a merge-based heuristic for Colorful Components. Our approach can solve the Colorful Components model of Wikipedia link correction optimally; while the Clique Partition-based ILP outperforms the other two approaches, the implicit hitting set is a simple approach that delivers competitive results. The merge-based heuristic is very accurate and outperforms the move-based one.

1 Introduction

Each entry in Wikipedia has links to the same entry in other languages. Sometimes, these links are wrong, while others are missing, since they are added and updated manually or by naïve bots. These errors can be detected by a graph model [3, 13, 14]: Each entry in a language corresponds to a vertex, and an interlanguage link corresponds to an edge. Then, ideally, a connected component in this graph would be a clique that corresponds to a single Wikipedia term in multiple languages, and, under the plausible assumption that for every language there is at most one Wikipedia entry on a particular term, each language should occur in a connected component only once. However, due to errors this is not the case. Our goal is to recover the correct terms by removing incorrect links and completing the resulting components. This can be done at minimum cost using a partitioning problem on a vertex-colored graph.

* Supported by project NANOPOLY (PITN-GA-2009–238700).
** Supported by DFG project PABI (NI 369/7-2).

> Colorful Components
> **Instance:** An undirected graph $G = (V, E)$ and a coloring of the vertices $\chi : V \to \{1, \ldots, c\}$.
> **Task:** Find a minimum-size edge set $E' \subseteq E$ such that in $G' = (V, E \setminus E')$, all connected components are *colorful*, that is, they do not contain two vertices of the same color.

We remark that the plain model naturally generalizes to an edge-weighted version and our solution strategies also apply to this.

Throughout the work, let $G = (V, E)$ where $n := |V|$ and $m := |E|$. To solve Colorful Components, we need to separate vertices of the same color. A *bad path* is a simple (i. e., cycle-free) path between two vertices of the same color.

Related work. Implicitly, Colorful Components has first been considered in a biological context as part of a multiple sequence alignment process, where it is solved by a simple min-cut heuristic [8]. Previously, we showed that it is NP-hard even in three-colored graphs with maximum degree six [4] and proposed an exact branching algorithm with proven worst-case running time bound and a merge-based heuristic which outperformed that of Corel et al. [8] on multiple sequence alignment data.

Avidor and Langberg [2] introduced Weighted Multi-Multiway Cut and provided results on its polynomial-time approximability (with non-constant approximation factors).

> Weighted Multi-Multiway Cut
> **Instance:** An undirected graph $G = (V, E)$ with edge weights $w : E \to \{x \in \mathbb{Q} : x \geq 1\}$ and vertex sets $S_1, \ldots, S_c \subseteq V$.
> **Task:** Find a minimum-weight edge set $E' \subseteq E$ such that in $G' = (V, E \setminus E')$ no connected component contains two vertices from the same S_i.

Colorful Components is the special case of Weighted Multi-Multiway Cut when the vertex sets form a partition.

A previous formalization of the Wikipedia link correction problem leads to a harder problem: it uses several separation criteria (instead of using only language data) and also allows to "ignore" the separation criterion for some vertices [13, 14]. The resulting optimization problem is a generalization of Weighted Multi-Multiway Cut. Since solving to optimality turned out to be too time-costly and non-scalable, a linear programming approach was followed [12, 13].

2 Solution Methods

We examine three approaches to finding optimal solutions for Colorful Components: One based on the implicit hitting set model by Moreno-Centeno and Karp [11, 15], and two based on integer linear programming (ILP) with row generation.

2.1 Implicit Hitting Set

Many NP-hard problems are naturally related to the well-known NP-hard Hitting Set problem, which is defined as follows:

> Hitting Set
> **Instance:** A ground set U and a set of *circuits* $S_1, \ldots, S_\ell$ with $S_i \subseteq U$ for $1 \leq i \leq \ell$.
> **Task:** Find a minimum-size *hitting set*, that is, a set $H \subseteq U$ with $H \cap S_i \neq \emptyset$ for all $1 \leq i \leq \ell$.

We can easily reduce Colorful Components to Hitting Set: The ground set U is the set of edges, and the circuits to be hit are all bad paths. Unfortunately, this can produce an exponentially-sized instance, and thus this approach is not feasible. However, we can model Colorful Components as an *implicit hitting set* problem [1, 6, 11, 15]: the circuits have an implicit description, and a polynomial-time oracle is available that, given a putative hitting set H, either confirms that H is a hitting set or produces a circuit that is not hit by H. In our case, the implicit description is simply the colored graph, and the oracle either returns a bad path that is not hit by H or confirms that H is a solution to Colorful Components.

Implicit hitting set models are useful for finding approximation algorithms [1, 6], but also for implementing exact solving strategies [15]. In the latter case, the approach is as follows. We maintain a list of circuits which have to be hit, initially empty. Then, we compute an optimal hitting set H for these circuits. If H is a feasible solution to the implicit hitting set instance, then it is also an optimal solution to Colorful Components. Otherwise, the oracle yields a bad path that is not destroyed by H. This bad path is added to the list of circuits, and we compute again a hitting set for this new list of circuits. This process is repeated until an optimal solution is found. The hitting set instances can be solved by using any Hitting Set solver as a black box; Moreno-Centeno and Karp [15] suggest an ILP solver, using a standard set-cover-constraint formulation.

As suggested by Moreno-Centeno and Karp [15], we use the following two tricks to speed up the computation. First, we initially solve each hitting set problem using a heuristic, and only use the ILP solver in case the oracle confirms that the heuristic solution gives a valid (but possibly non-optimal) solution for Colorful Components. Second, instead of adding only one new circuit in each iteration, we greedily compute a set of disjoint shortest bad paths that are added to the circuit set.

2.2 Hitting Set ILP formulation

Moreno-Centeno and Karp [15] mention that their approach is related to column (variable) generation schemes for ILP solvers. Possibly even more straight-forward, we can solve any implicit hitting set problem with an ILP solver by a row (constraint) generation scheme (also called “lazy constraints” in the well-known CPLEX solver). For this, we declare binary variables $h_1, \ldots, h_{|U|}$, where the value of h_i is to indicate whether the ith element of U (under some arbitrary order) is in H. The objective is to minimize $\sum_i h_i$. We then start the branch-and-bound process with an empty constraint set and, in a callback, query the oracle for further constraints once an integer feasible solution is obtained. If new constraints are generated, they are added to the problem, cutting off some parts of the search tree. Otherwise, we have found a valid solution. Note that adding lazy constraints is different from adding cutting planes, since cutting planes are only allowed to cut off fractional solutions that would not be integer feasible, whereas lazy constraints can also cut off integer feasible solutions.

More concretely, for Colorful Components, we have a variable $d_{uv}, u < v$, for each $\{u, v\} \in E$, where $d_{uv} = 1$ indicates that edge $\{u, v\}$ gets deleted. We then want to minimize $\sum_{e \in E} d_e$. The oracle deletes all edges $\{u, v\}$ with $d_{uv} = 1$, and then looks for a bad path $u_1, \ldots, u_l$. If it finds one, it yields the *path inequality*

$$\sum_{i=1}^{l-1} d_{u_i u_{i+1}} \geq 1. \tag{1}$$

We could hope that this process is more effective than the general implicit hitting set approach which uses an ILP solver as a black-box solver, since constraints are generated early on without the need for the solver to optimally solve subproblems that yield solutions that are not globally feasible.

The main disadvantage of this approach, compared to the implicit hitting set formulation, is that it requires a solver-specific implementation; further, some ILP solvers such as Coin CBC 2.7 or Gurobi 4.6 do not support adding lazy constraints without starting the solving process from scratch (the recently released Gurobi 5.0 adds this feature).

2.3 Clique Partitioning ILP formulation

It is known for a long time (e. g. [7]) that multicut problems can be reduced to Clique Partitioning. In this problem, vertex pairs are annotated as being similar or as being dissimilar, and the goal is to find a partition of the vertices that maximizes consistency with these annotations. We model the partition of the vertices as a *cluster graph*, that is, a graph where every connected component is a clique. The formal problem definition is then as follows:

Clique Partitioning
Instance: A vertex set V with a weight function $\delta : \binom{V}{2} \to \mathbb{Q}$.
Task: Find a cluster graph (V, E) that minimizes $\sum_{\{u,v\} \in E} \delta(u, v)$.

Herein, $\delta(u, v)$ denotes the *dissimilarity* between u and v.

To obtain a Clique Partitioning instance from a Colorful Components instance, we set

$$\delta(u, v) = \begin{cases} \infty & \text{if } \chi(u) = \chi(v), \\ -1 & \text{if } \{u, v\} \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

A component in a feasible solution for this Clique Partitioning instance cannot contain more than one vertex of a color, since the component is a clique and the two vertices would be connected, incurring a cost of ∞. Thus, the solution also is a feasible solution for Colorful Components, and the cost is the number of edges between components, and therefore equals the number of edges that need to be deleted for Colorful Components.

There is a well-known ILP formulation of Clique Partitioning [9, 17], which we can adapt for Colorful Components. It has been successfully implemented and augmented with cutting planes [5, 9, 16]. It uses binary variables e_{uv} for $u, v \in V, u < v$, where $e_{uv} = 1$ iff the edge $\{u, v\}$ is part of the solution cluster graph. Cluster graphs are exactly those graphs that do not contain a P_3 as induced subgraph, that is, three distinct vertices u, v, w with $\{u, v\} \in E$ and $\{v, w\} \in E$ but $\{u, w\} \notin E$. Thus, we can ensure that the graph is a cluster graph by avoiding a P_3 for each possible triple of vertices $u < v < w \in V$:

$$e_{uv} + e_{vw} - e_{uw} \leq 1 \tag{3}$$

$$e_{uv} - e_{vw} + e_{uw} \leq 1 \tag{4}$$

$$-e_{uv} + e_{vw} + e_{uw} \leq 1 \tag{5}$$

We can shrink the ILP by substituting $e_{uv} = 0$ for $u \neq v \in V, \chi(u) = \chi(v)$. Finally, the objective is to minimize $\sum_{\{u,v\} \in E} \delta(u, v) e_{uv}$.

Compared to the formulation from Section 2.2, an advantage of this formulation is that it has only polynomially many constraints, as opposed to exponentially many, and therefore can often be stated explicitly. However, the number of constraints is $3\binom{n}{3} = O(n^3)$, and thus can get easily too large for memory. Therefore, we also implement here a row generation scheme. We find violated inequalities by a simple brute-force search. When finding a violated inequality involving vertices u, v, w, we add all three inequalities (3)–(5), since we found this to be more efficient in our experiments.

2.4 Cutting Planes

As mentioned, the effectiveness of ILP solvers comes from the power of the relaxation. We enhance this by adding *cutting planes*, which are valid constraints that cut off fractional solutions. This scheme is called *branch-and-cut*. As demonstrated in Section 3, this addition to generic Clique Partitioning or Hitting Set approaches is necessary to obtain competitive performance.

First, since for both the Hitting Set and the Clique Partitioning formulation we are using row generation, we can check if already a fractional solution violates a problem-defining constraint. This allows to improve the relaxation and to cut off infeasible solutions earlier in the search tree. For the hitting set formulation, violated constraints can be found by a modified breadth-first search from each vertex that considers the current variable values, and for the clique partitioning model, violated constraints can be found by a simple brute-force search.

Chopra and Rao [7] suggest several cutting planes for Multiway Cut, the special case of Multi-Multiway Cut with $c = 1$. Each color of a Colorful Components instance induces a Multiway Cut polytope. The Colorful Components polytope is thus an intersection of several multiway cut polytopes. Therefore, these cutting planes are valid for Colorful Components, too. We present them here for the Clique Partitioning formulation.

Let $T = (V_T, E_T)$ be a subgraph of G that is a tree such that all leaves L of the tree have color c, but no inner vertex has. Then the inequality

$$\sum_{uv \in E_T} e_{uv} \leq (|E_T| - |L|) + 1 \tag{6}$$

is called a *tree inequality*. Note that for $|L| = 2$, we get a path inequality (1). There are exponentially many tree inequalities. Therefore, we consider only tree inequalities with one (star inequalities) or two internal vertices. We can also apply these cuts for the Hitting Set formulation: we need to substitute $e_{uv} = 1 - d_{uv}$ and restrict the sums to edges present in the graph. In our implementation of the Hitting Set row generation scheme, we add all initially violated star inequalities at once. Note that this already covers all length-2 bad paths.

2.5 Heuristics

One advantage of having an algorithm that is able to solve large-scale instances optimally is that we can evaluate heuristics more precisely. Our merge heuristic was shown to be quite successful [4], and as greedy vertex moving was shown to perform better than a merge strategy [10] for density-based partitioning, we wanted to compare the two. We examine our previous heuristic [4], which outperformed the one proposed by Corel et al. [8]. For completeness, we briefly recall this greedy heuristic [4]. The idea is to repeatedly merge the two vertices "most likely" to be in the same component. During the process, we immediately delete edges connecting vertices with identical colors. Thus, we can determine the *merge cost* of two vertices u and v as the weight of the edges that would need to be deleted in this way when merging u and v. The *cut cost* is an approximation of the minimum cut between u and v obtained by looking only at their common neighbors. We then repeatedly merge the endpoints of the edge that maximizes cut cost minus merge cost.

The second heuristic we consider is *greedy vertex moving* as proposed by Görke et al. [10]. Here we start with singleton clusters, that is, every cluster contains exactly one vertex. Then, we consider all possible ways of moving one vertex from a cluster to another. Of these possibilities, we greedily perform the one that decreases the number of inter-cluster edges the most without violating the colorfulness condition. After no improvement is possible anymore, clusters are merged into vertices and the procedure is applied recursively.

3 Experiments

We performed experiments both to evaluate the model and to compare the three solution approaches. The ILP approaches were implemented in C++ using the CPLEX 12.4 ILP solver. The experimental data and the code are available at http://fpt.akt.tu-berlin.de/colcom/. The test machine is a 3.6 GHz Intel Xeon E5-1620 with 10 MB L3 cache and 64 GB main memory, running under Debian GNU/Linux 7.0. Only a single thread was used.

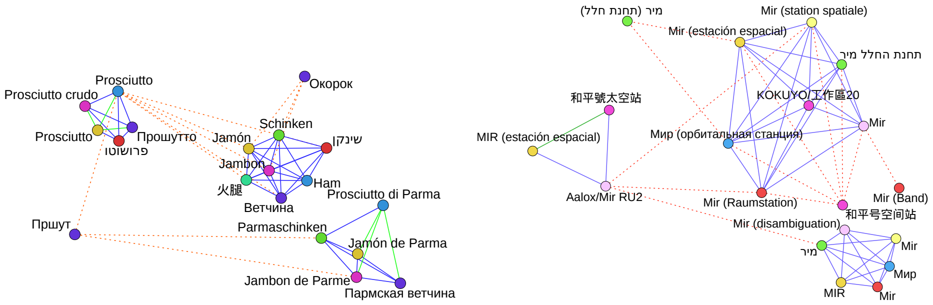
Each connected component is solved separately. We use data reduction as described by [4] before starting the solver. This actually yields an instance of the more general weighted MULTI-MULTIWAY CUT problem. Adapting the ILP formulations above to this problem is straightforward. We further use the result of the merge-based heuristic (Section 2.5) as *MIP start* (that is, we pass this solution to the ILP solver such that it can start with a good upper bound).

3.1 Medium-Sized Wikipedia Graphs

To construct the Wikipedia interlanguage graph, we downloaded the freely-available Wikipedia interlanguage links and page data dumps from January 9th, 2012. We chose a set of seven languages: English, Spanish, German, French, Russian, Chinese and Hebrew. We then created the graph as described in the introduction, with vertices as pages and a link between two pages if one has an interlanguage link to the other. As suggested by de Melo and Weikum [13], we weigh the edges as follows: If two pages link to each other, the edge receives a weight of 2. Otherwise, the weight is 1. The graph contains 4,090,160 vertices and 9,666,439 edges in 1,332,253 connected components. The largest connected component has size 409. Of these components, 1,252,627 are already colorful.

We then found the colorful components using the CLIQUE PARTITIONING algorithm from Section 2.3, in 54 seconds. The merge-based heuristic obtained a solution that was 0.22% off the optimum, taking 38 seconds, and the greedy vertex moving was 0.90 % off, taking 37 seconds. Note that our implementations of the heuristics have not been optimized for speed.

The cost of the optimal solution is 188,843, deleting 184,759 edges. After removing the edges of the solution we had 1,432,822 colorful components. We obtained 1,355,641 colorful components of size > 1, each corresponding to an entry in several languages. A little less than half of the colorful components contain two vertices, 20% contain three vertices, $11\%, 6\%, 3\%, 2\%$ contain four, five, six and seven vertices, respectively. The remaining components are singletons. In each colorful component that is not already a clique, two vertices that are not already connected by an edge represent two pages in different languages that should have a new inter-language link between them. Overall, we found 52,058 such new links. To get an idea of the correctness of these links we looked at the Hebrew and English pages and manually checked the new links between them. For example, we identified missing links between "data compression" and its Hebrew counterpart, and "scientific literature" and the equivalent Hebrew entry, which was previously linking to the less fitting "academic publishing".



(a) A single connected component in the Wikipedia graph, disambiguating different types of pork. The English page "Prosciutto" is connected both to the correct "Prosciutto" cluster and to the "Ham" cluster. The Colorful Components algorithm separates the two correctly.

(b) A single connected component in the Wikipedia graph, regarding the term "MIR". The algorithm successfully separates the cluster of entries corresponding to the disambiguation of the term from the cluster centering on the MIR space station. The outlier for the band MIR is now also disconnected.

Fig. 1. Two connected components in the Wikipedia graph. Green edges have been inserted and dotted red edges have been deleted by the algorithm.

Figures 1(a) and 1(b) demonstrate results of the algorithm. In both cases the algorithm successfully separates clusters representing related, but not identical terms and identifies outliers.

3.2 Large-Scale Wikipedia Graphs

To test our fastest ILP formulation (Clique Partitioning) and the heuristics on even larger inputs, we downloaded Wikipedia interlanguage link data for the largest 30 languages[3] on 7 June 2012. To decrease noise, we excluded user pages and other special pages. The resulting instance has 11,977,500 vertices and 46,695,719 edges. Of the 2,698,241 connected components, 225,760 are not colorful, the largest of which has 1,828 vertices and 14,403 edges. The instance can be solved optimally in about 80 minutes (we cannot give a more precise figure since because of memory constraints, we had to run our implementation on a different machine that was also loaded with other tasks). In the solution, 618,660 edges are deleted, and the insertion of 434,849 can be inferred. The merge-based heuristic has an error of 0.81 %. Solving the largest component takes 182 seconds; 10.2 % of the edges are deleted. The merge-based heuristic takes 13.4 seconds, with 1.15 % error. The languages in the component are similarly distributed as in the overall graph. It contains mostly terms related to companies, in particular different legal forms of these, and family relationships. We noted that many inconsistencies have been introduced by bots that aim to fill in "missing"

[3] http://meta.wikimedia.org/wiki/List_of_Wikipedias

translations; for example, the Hungarian word “Részvény” (stock) is wrongly linked to the term for “free float” in many languages.

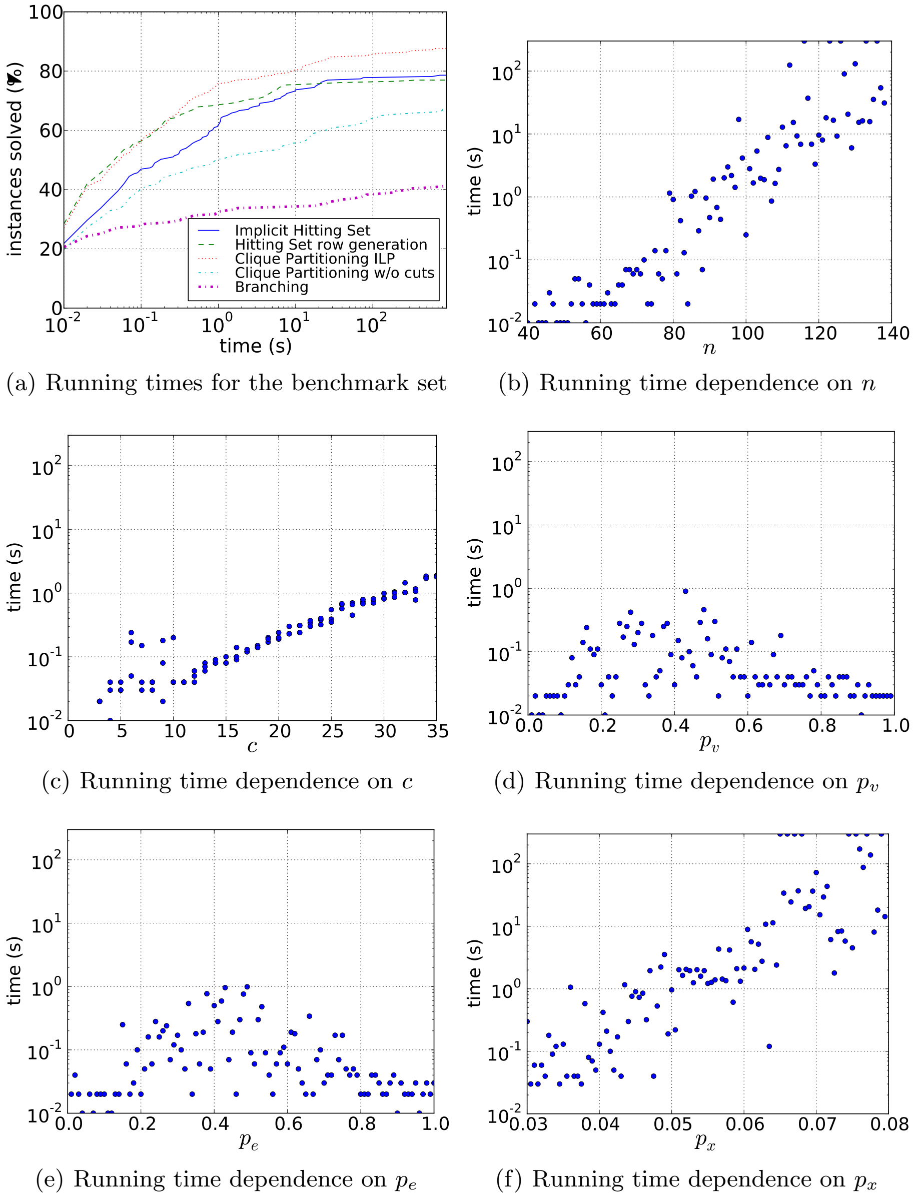
3.3 Random Graphs

To compare the performance of our approaches, we generated a benchmark set of random instances. The model is the recovery of colorful components that have been perturbed. More precisely, the model has five parameters: c is the number of colors; n is the number of vertices; p_v is the probability that a component contains a vertex of a certain color; p_e is the probability that between two vertices in a component there is an edge; p_x is the probability that between two vertices from different components there is an edge.

Clearly, for the instances to be meaningful, p_e must be much higher than p_x. We first generate a benchmark set of 243 instances with parameters similar to those of the largest connected components in the 7-language Wikipedia instance. Note that since each instance models a connected component, they are much smaller than a typical real-world instance.

Based on the parameters corresponding to the largest connected components in the Wikipedia instance, we choose the parameters as follows: $c \in \{3, 5, 8\}$, $n \in \{60, 100, 170\}$, $p_v \in \{0.4, 0.6, 0.9\}$, $p_e \in \{0.4, 0.6, 0.9\}$, and $p_x \in \{0.01, 0.02, 0.04\}$. In Fig. 2(a), we compare the running times for the three approaches and additionally the branching algorithm from [4], with a time limit of 15 minutes. The branching algorithm is clearly not competitive. Among the ILP based approaches, the Clique Partitioning formulation eventually comes out as a winner. All instances with $n = 60$ are solved, and only 4 of the $n = 100$ instances remain unsolved, all of which have $p_x = 0.04$. The performance of the row generation scheme is somewhat disappointing, solving less instances than the implicit hitting set formulation. One possible reason is that for the implicit hitting set formulation, the solver is able to employ its presolve functions to simplify the instance. Further tuning and application of cutting planes might give the row generation scheme an advantage, though.

We now compare the effect of varying a single parameter, starting with the base parameters $n = 70$, $c = 6$, $p_v = 0.6$, $p_e = 0.7$, and $p_x = 0.03$. We set a timeout of 5 minutes. In Fig. 2(b), the exponential growth of the running time when increasing the instance size is clearly visible. This is as expected for an exact approach to an NP-hard problem. In Fig. 2(c), we vary the number c of colors. The running time grows with more colors, but remains manageable. The parameter p_v does not seem to have a large effect on running times (Fig. 2(d)); the approach copes well even with components with many missing vertices. For the parameter p_e (Fig. 2(e)), we note lower running times for high values. This matches intuition, since dense clusters should be easier to identify. The running time is also lower for small values; this can probably be explained by the fact that such instances have overall very few edges. Finally, we see that the parameter p_x, which models the number of “errors” in the instance, has a large influence on running time (Fig. 2(f)); in fact, the running time also seems to grow exponentially with this parameter.



(a) Running times for the benchmark set

(b) Running time dependence on n

(c) Running time dependence on c

(d) Running time dependence on p_v

(e) Running time dependence on p_e

(f) Running time dependence on p_x

Fig. 2. Running times for synthetic Colorful Components instances

Finally, we examine the performance of the heuristics (Section 2.5) on the benchmark set for those 213 instances where we know the optimal solution. The maximum running time for an instance is 0.4 s for both heuristics. The merge-based heuristic finds an optimal solution for 124 instances; the average error is 0.86 % and the maximum 12.5 %. The move-based heuristic finds an optimal solution for 55 instances; the average error is 4.9 % and the maximum 38.7 %.

Discussion. The most critical parameter that determines whether the exact methods can successfully be employed is the amount of inter-cluster edges (that is, the solution size), since it determines both the size n of connected components and the parameter p_x. If this value is small enough, then even very large instances like the Wikipedia interlanguage network can be solved optimally. Otherwise, the merge-based heuristic provides excellent results typically very close to the optimum.

Among the exact approaches, the Clique Partitioning ILP formulation performs better than the implicit hitting set approach, but its implementation is tied to a specific solver (in our case, the proprietary CPLEX), while the implicit hitting set approach can easily be adapted to any ILP solver including free software solvers. Thus, there are use cases for both, while the Hitting Set row generation does not seem like a good option in its current form.

Similar to our previous results for multiple sequence alignment [4], the merge-based heuristic gives an excellent approximation here. It also clearly outperforms a move-based approach, in contrast to the results of Görke et al. [10] for density-based clustering. A possible explanation is that the merge-based heuristic already takes the color constraints into account when determining the cost of a modification, and not only for its feasibility.

4 Outlook

There are several ways the methods presented here could be improved. For the implicit hitting set, there are many further ways to tune it [15]. For the ILPs, it would be interesting to find cutting planes that take vertices of more than one color into account. In ongoing work, we experimented with a column generation approach based on the Clique Partitioning model, using a greedy heuristic and an ILP formulation for solving the column generation subproblem. While on the synthetic data it is slower than the other ILP-based approaches with a time limit of 10 seconds, it can solve almost as many instances as the fastest approach after 15 minutes. Thus, it seems to be a good candidate for solving even larger-scale problems. As a next natural step, one should also see whether our mathematical programming solving methods for Colorful Components which are based on Weighted Multi-Multiway Cut formulations extend to applications where one actually *needs* to solve the more general Weighted Multi-Multiway Cut. For instance, cleansing of taxonomies [12] would be a natural candidate.

Concerning applications and modeling, there are several ways to expand our results. First, we currently only demand a cluster to be a connected subgraph;

further restrictions on its density might be useful. Also, for some applications the conditions on colors in a component might be relaxed, for example by allowing a constant number of duplicates per component. Finally, finding further applications would be interesting. We briefly sketch one candidate application here. Consider a graph where each vertex corresponds to a user profile in a social network, two profiles are adjacent when they are similar, and the color of a vertex is the network (Twitter etc.). Then, Colorful Components could be used to identify groups of profiles that correspond to the same natural person, assuming that every person has at most one profile in each network.

References

[1] M. V. Ashley, T. Y. Berger-Wolf, W. Chaovalitwongse, B. DasGupta, A. Khokhar, and S. Sheikh. An implicit cover problem in wild population study. *Discrete Mathematics, Algorithms and Applications*, 2(1):21–31, 2010.

[2] A. Avidor and M. Langberg. The multi-multiway cut problem. *Theoretical Computer Science*, 377(1–3):35–42, 2007.

[3] Ł. Bolikowski. Scale-free topology of the interlanguage links in Wikipedia. Technical Report arXiv:0904.0564v2, arXiv, 2009.

[4] S. Bruckner, F. Hüffner, C. Komusiewicz, R. Niedermeier, S. Thiel, and J. Uhlmann. Partitioning into colorful components by minimum edge deletions. In *Proc. 23rd CPM*, volume 7354 of *LNCS*, pages 56–69. Springer, 2012.

[5] S. Böcker, S. Briesemeister, and G. W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 60(2):316–334, 2011.

[6] K. Chandrasekaran, R. M. Karp, E. Moreno-Centeno, and S. Vempala. Algorithms for implicit hitting set problems. In *Proc. 22nd SODA*, pages 614–629. SIAM, 2011.

[7] S. Chopra and M. R. Rao. On the multiway cut polyhedron. *Networks*, 21(1): 51–89, 1991.

[8] E. Corel, F. Pitschi, and B. Morgenstern. A min-cut algorithm for the consistency problem in multiple sequence alignment. *Bioinformatics*, 26(8):1015–1021, 2010.

[9] M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1–3):59–96, 1989.

[10] R. Görke, A. Schumm, and D. Wagner. Experiments on density-constrained graph clustering. In *Proc. 2012 ALENEX*, pages 1–15. SIAM, 2012.

[11] R. M. Karp. Heuristic algorithms in computational molecular biology. *Journal of Computer and System Sciences*, 77(1):122–128, 2011.

[12] T. Lee, Z. Wang, H. Wang, and S. Hwang. Web scale taxonomy cleansing. In *Proceedings of the VLDB Endowment*, volume 4, pages 1295–1306, 2011.

[13] G. de Melo and G. Weikum. Untangling the cross-lingual link structure of Wikipedia. In *Proc. 48th ACL*, pages 844–853. ACM, 2010.

[14] G. de Melo and G. Weikum. MENTA: inducing multilingual taxonomies from Wikipedia. In *Proc. 19th CIKM*, pages 1099–1108. ACM, 2010.

[15] E. Moreno-Centeno and R. M. Karp. The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment. *Operations Research*, 2013. To appear.

[16] M. Oosten, J. H. G. C. Rutten, and F. C. R. Spieksma. The clique partitioning problem: Facets and patching facets. *Networks*, 38(4):209–226, 2001.

[17] S. Régnier. Sur quelques aspects mathématiques des problèmes de classification automatique. *I.C.C. Bulletin*, 4:175–191, 1965.