An Efficient Branch and Cut Algorithm to Find Frequently Mutated Subnetworks in Cancer

Anna Bomersbach[1], Marco Chiarandini[1], and Fabio Vandin[1,2,3(✉)]

[1] Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark
anna.bomersbach@gmail.com, marco@imada.sdu.dk
[2] Department of Information Engineering, University of Padova, Padova, Italy
vandinfa@dei.unipd.it
[3] Department of Computer Science, Brown University, Providence, USA

Abstract. Cancer is a disease driven mostly by somatic mutations appearing in an individual's genome. One of the main challenges in large cancer studies is to identify the handful of driver mutations responsible for cancer among the hundreds or thousands mutations present in a tumour genome. Recent approaches have shown that analyzing mutations in the context of interaction networks increases the power to identify driver mutations.

In this work we propose an ILP formulation for the exact solution of the combinatorial problem of finding subnetworks mutated in a large fraction of cancer patients, a problem previously proposed to identify important mutations in cancer. We show that a branch and cut algorithm provides exact solutions and is faster than previously proposed greedy and approximation algorithms. We test our algorithm on real cancer data and show that our approach is viable and allows for the identification of subnetworks containing known cancer genes.

Keywords: Cancer mutations · Branch and cut · Combinatorial optimization · Network analysis

1 Introduction

Recent advances in DNA sequencing technologies have allowed the study of cancer genomes at an unprecedented level of detail. In particular, it is now possible to measure all *somatic mutations*, changes in the DNA arising during the lifetime of an individual and causing the disease, in a large number of cancer patients [12,28]. These large cancer studies have shown that each individual tumour harbours hundreds or thousands somatic mutations, with two tumours showing a large diversity in the complement of somatic mutations they exhibit [9,27]. This phenomenon is commonly referred to as (intertumor) cancer heterogeneity.

M. Frith and C.N.S. Pedersen (Eds.): WABI 2016, LNBI 9838, pp. 27–39, 2016.
DOI: 10.1007/978-3-319-43681-4_3

Cancer heterogeneity is explained by the fact that only a handful of all the somatic mutations in a cancer genome are *driver* mutations related to the diseases, while the majority of mutations are *passenger* mutations not related to cancer progression and development. Moreover, driver mutations target regulatory and signaling *pathways*, groups of interacting genes that perform specific functions in the cell [10,26] and that may be altered by mutating any of the genes in the group. Therefore, to identify all driver mutations and the genes they affect one cannot focus on genes in isolation, but has to study mutations in the context of interaction networks [5].

In recent years, several methods have been proposed to identify significantly mutated pathways in cancer [21]. Some of these methods work on known pathways [4], thus limiting our ability to identify novel pathways as well as subnetworks connecting two pathways that are important for cancer. Other methods identify significantly mutated pathways by combining mutation data with a large protein-protein interaction network [14,15,18,23,25]. A common formulation is to look for connected subnetworks that are mutated in a large number of patients, that is equivalent to identifying connected subnetworks whose vertices *cover* a large number of elements (i.e., patients) from a universe. In particular [25] defined the connected maximum coverage problem (CMCP) as finding a connected subnetwork of cardinality k that covers the maximum number of patients, proven to be NP-hard in [25] where an approximation algorithm was also presented.

In this paper, we propose an integer linear programming formulation for CMCP. Our formulation draws from an analogous formulation for Steiner tree problems that have been the object of a recent DIMACS challenge [13]. In particular, the connectivity constraint leads to an exponential number of constraints that we handle within a branch and cut framework. We show that our algorithm allows for the identification of the optimal solution of CMCP on real cancer datasets, and that the identified solutions cover more patients compared to previously proposed heuristic approaches or approximation algorithms. We also show that the subnetworks identified by our approach have higher statistical significance, estimated through permutation testing, compared to solutions found by the approximation algorithm. We generalize our formulation to the weighted version of the problem, and show that our branch and cut strategy can be used to solve this formulation as well, and also show that our approach identifies subnetworks of genes known to be associated with cancer.

Related Work. The computational problem of identifying connected subnetworks with vertices covering a large number of elements have been studied in bioinformatics [14,15,24,25] as well as in wireless network design [16]. As mentioned above, [25] studied the CMCP and provided an approximation algorithm for its solution (see Sect. 2.1); [14,15,24] studied related, but different, problems. In gene expression studies, [24] studied the problem of finding the smallest connected subnetwork such that at least k genes in the subnetwork are differentially expressed in all patients but at most ℓ. [14,15] study the problem of finding the minimum cost collection of modules (i.e., subgraphs) covering each patient at least k times, where a patient is covered by a module if at least one gene in the module is altered in the

patient. The cost of a collection of modules is a function of the size of the modules and other pairwise properties of the genes in the module (e.g., their distance in a network; the degree of exclusivity of alterations).

The CMCP has some similarity with the cardinality constrained Maximum Weight Connected Subgraph Problem [1,6], that asks to find a connected subgraph with maximum total weight in a node-weighted graph, and with the prize-collecting Steiner Tree Problem [8,13,19], that asks to find a subtree of minimum costs that spans all vertices from a set of terminals. Different ILP formulations for these problems have recently been studied both theoretically and in practice [1,6,29]. The main issue from an ILP perspective is modeling the connectivity requirement. In the formulations of [1,6], the connectivity constraints are formulated by means of a root node and a generalized form of node separators. With this approach, an additional set of node variables is needed to locate the root. Later [8] proposes a thinned formulation that does not need a root and variables to locate it and that uses node separators in non-generalized form. Both formulations exhibit an exponential number of connectivity constraints; therefore, branch and cut (B&C) algorithms have been used to solve these models. In this approach connectivity constraints are not explicitly declared, rather they are introduced during the search when they are needed. The B&C algorithm by [1,6] finds violated connectivity cuts by identifying a minimum cut in a support digraph. The B&C algorithm by [8] uses a lazy approach in which violated constraints are only searched when an integer solution is found and employs a linear time algorithm to discover minimal node separators that are facet-defining. [3] introduced a flow based, polynomial size formulation for connectivity constraints for the problem of finding colorful connected subgraphs.

We note that while the connectivity requirement of CMCP is the same as the connectivity requirements in the Maximum Weight Connected Subgraph Problem and the prize-collecting Steiner Tree Problem, the latter two problems have objective functions that are additive in the vertices chosen in the solution (and also in the edges for the prize-collecting Steiner Tree Problem), while the objective function for the CMCP is the more complicated coverage function, that is a submodular set function [16].

2 Model and Algorithms

We are given a graph $G = (V, E)$, with vertices $V = \{1, \ldots, n\}$ representing genes and edges E representing interactions among genes (or the associated proteins). Let P denotes the set of patients for which mutations have been assayed. Let $P_i \subseteq P$ be the set of patients in which gene $i \in V$ is mutated. We say that a patient $j \in P$ is *covered* by a subset of vertices $S \subseteq V$, if there exists at least one vertex v in S such that $j \in P_v$.

Our goal is to identify connected subgraphs of G that are mutated in a large number of patients, where a subgraph is mutated in a patient if at least one of the vertices in the subgraph covers the patient. More formally, we consider the following problem, defined in [25].

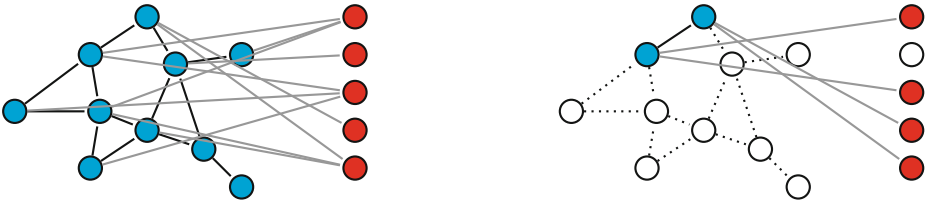

Fig. 1. Left: an instance of CMCP, with blue vertices and black edges representing G, red vertices representing patients P and gray lines linking patients to their mutated genes. Right: an optimal solution (in blue) to the instance on the left for $k = 2$. Patients covered by the optimal solution are in red. (Color figure online)

Connected Maximum Coverage Problem (CMCP). Given a graph G defined on a set of n vertices V, an integer $k > 0$, a set P, a family of subsets $\mathcal{P} = \{P_1, \ldots, P_n\}$ where for each $i, P_i \subseteq P$ is associated with $i \in V$, find the connected subgraph $S^* \subseteq G$ with k vertices that maximizes the coverage $|\cup_{i \in S^*} P_i|$.

Figure 1 shows an instance of CMCP. If G is a complete graph, the connected maximum coverage problem is the maximum coverage problem [11], where, given a set U of elements, a family of subsets $\mathcal{F} \subset 2^U$, and a value k, one needs to find a collection of k sets in $\mathcal{F}$ that covers the maximum number of elements in U. The maximum coverage problem is NP-hard [11], thus, the connected maximum coverage problem is NP-hard for a general graph and even for star graphs [25].

Preprocessing. Only connected components of G of size $\geq k$ need to be considered. The problem can be solved for each of those components in turn, returning the best solution found. Nodes $v \in V$ that cannot be in an optimal solution are removed using the following rules: (i) v has degree 1, $P_v = \emptyset$, and after removal the number of nodes in the connect component is $\geq k$; (ii) there is a node $v' \neq v$ whose set of neighbors is a superset of the set of neighbors of v and the set $P_{v'}$ is a superset of the set P_v.

The rest of the section is organized as follows: Sect. 2.1 reviews previously proposed algorithms for the CMCP; Sect. 2.2 presents our ILP formulation for the CMCP and the corresponding branch and cut algorithm; Sect. 2.3 extends the ILP to a weighted version of the CMCP; and Sect. 2.4 describes the permutation test used to assess the statistical significance of the solutions.

2.1 Previous Methods: Approximation Algorithm and Greedy Algorithms

In this section we present previously proposed methods as well as simple greedy algorithms for the CMCP. [7] proposed a polynomial time $1/(cr)$-approximation algorithm, where $c = (2e-1)/(e-1)$ and r is the radius of optimal solution in the graph. The algorithm starts by computing and storing all pairs shortest paths. Then the algorithm finds a solution starting from each node v of the graph, and at the end reports the best solution found. Given the current solution S obtained

starting from vertex v, the algorithm augments it by adding the (shortest) path to a vertex u that maximizes the ratio between the number of newly covered patients and the number of new vertices added to the solution, while keeping the number of vertices in the solution $\leq k$. We also implemented two variants of the approximation algorithm that do not compute all pairs shortest paths and differ in the way they define *candidate paths* to extend the current solution. The first variant (`bfs`) performs a BFS (of depth $\leq k$) starting from the node v from which the solution is grown. The second variant (`ratio`) considers *all* paths (not only shortest ones) among pairs of vertices, and it does not guarantee to run in polynomial time but has been shown to run efficiently on real datasets [7].

We also consider a simple greedy algorithm that builds a solution starting from each vertex v and reports the best solution found. The algorithm builds a solution by always adding to the current solution S the neighboring node providing the maximum increase in the coverage.

2.2 ILP Formulation and Branch and Cut Algorithm

Our ILP formulation for CMCP is analogous to a recent formulation [8] for the prize-collecting Steiner Tree problem. It involves only node variables and is based on node separator inequalities, some of which can be proved to be facet defining for the connected subgraph polytope [29].

Given the graph $G = (V, E)$ and two distinct nodes h and ℓ from V, a subset of nodes $N \subseteq V \setminus \{h, \ell\}$ is an (h, ℓ) *(node) separator* if and only if after removing N from V there is no path between h and ℓ in G. Let $\mathcal{N}(h, \ell)$ denote the family of all (h, ℓ) separators. A separator $N \in \mathcal{N}(h, \ell)$ is minimal if $N \setminus \{i\}$ is not an (h, ℓ) separator, for any $i \in N$. We use binary coefficients a_{ij} for $i \in V$ and $j \in P$ to indicate whether i covers j (i.e., gene i is mutated in patient j), that is, $a_{ij} = 1$ if $j \in P_i$, and $a_{ij} = 0$ otherwise. Let x_i for $i \in V$ be binary variables such that $x_i = 1$ if the vertex i is in the solution $S \subseteq V$ and $x_i = 0$ otherwise, and let y_j for $i \in V$ be binary variables such that $y_j = 1$ if patient j is covered by S and $y_j = 0$ otherwise. We formulate the CMCP as an ILP as follows:

$$\max \sum_{j \in P} y_j \tag{1}$$

$$\sum_{i \in V} x_i = k \tag{2}$$

$$\sum_{i \in V} a_{ij} x_i \geq y_j \qquad \forall\, j \in P \tag{3}$$

$$\sum_{i \in N} x_i \geq x_h + x_\ell - 1 \qquad \forall\, h, \ell \in V, h \neq \ell, \forall\, N \in \mathcal{N}(h, \ell) \tag{4}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V \tag{5}$$

$$y_j \in \{0, 1\} \qquad \forall j \in P \tag{6}$$

Constraints (2) impose that exactly k nodes of V are in the solution S. Constraints (3) ensure that the variables y_j are set to one only when the corresponding $j \in P$ is covered. Constraints (4) are the node separator constraints for the connectivity requirement. They ensure that for any pair of nodes h, ℓ in S there is a path in the graph induced by S, i.e., for any node separator $N \in \mathcal{N}(h, \ell)$ at least one node in N must also be selected. [29] shows that the constraints (4) are facets defining for the connected subgraph polytope if and only if N is a minimal node separator separating h and ℓ.

The Branch and Cut Algorithm. Our B&C algorithm (`B&C`) is analogous to the one for the Steiner tree problem in [8]. The connectivity cuts (4) are treated as lazy constraints, that is, they are not explicitly represented in the initial ILP model but they are introduced only when an integer solution that violates any of these constraints is found. `B&C` starts by including only a special case of inequalities (4), i.e., $x_i \leq \sum_{\ell \in V:(i,\ell) \in E} x_\ell$ for all $i \in V$. They state that a node, if selected, must have a neighbor also selected. At any node of the branch and bound tree an integral solution $\tilde{\boldsymbol{x}}$ to the model (1)–(6) with a subset of the constraints (4) gives a set of nodes $\tilde{S} = \{i \in V \mid \tilde{x}_i = 1\}$. If the solution is not feasible with respect to the full model, then in the subgraph of G induced by $\tilde{S}$ there are disjoint connected components. Let C_h and C_ℓ be two such components, containing the nodes h and ℓ, respectively. Then, the linear time algorithm from [8] reported in Fig. 2 and exemplified in Fig. 3 finds the minimal node separator that must be added to the model. In our implementation, for an integer solution with m disjoint connected components we find the minimal node separator for all $m \times (m-1)$ combinations and add all corresponding constraints. `B&C` terminates when an integer solution that does not violate any lazy constraint and whose value is proven optimal is found.

Function findMinNodeSeparator$(G, \tilde{S}, \{h, \ell\} \in \tilde{S}, C_h)$
  $A(C_h) \leftarrow$ neighbors of nodes of C_h in G
  $G' \leftarrow G$ with all edges between vertices in $C_h \cup A(C_h)$ removed
  $R_\ell \leftarrow$ nodes that can be reached from ℓ in G'
  return $N = A(C_h) \cap R_\ell$

Fig. 2. Linear time algorithm for finding a minimal node separator N. The input is G, two nodes h and ℓ of an infeasible solution $\tilde{S}$, and the connected component C_h of the subgraph of G induced by $\tilde{S}$ containing h and not containing ℓ.

2.3 Weighted Model

We extend the ILP formulation to the case where for each (gene) vertex $i \in V$ and each patient $j \in P$ we have a weight w_{ij} that we gain when i is used to cover j; the objective is to maximize the weight of the covered patients with at most one gene that can be picked to cover a patient. Without loss of generality

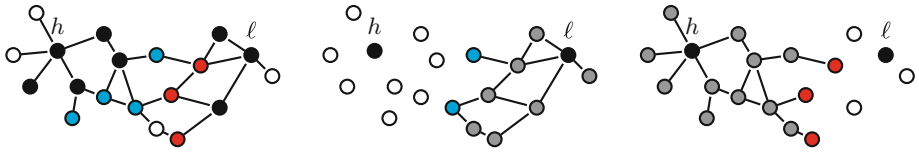

Fig. 3. Left: example of disconnected solution $\tilde{S}$ (black nodes). Blue nodes are nodes $A(C_h)$, neighbors of C_h, and constitute a non minimal node separator. Middle: grey and blue nodes represent the nodes in R_ℓ that are reachable from ℓ in G' and the blue nodes constitute the minimal node separator. Right: the red nodes represent the minimal node separator determined repeating the same procedure for component C_ℓ. (Color figure online)

we assume that for all $i \in V$ and all $j \in P : w_{ij} \leq 1$ (weights can be normalized by dividing them by the highest value). We introduce binary variables z_{ij} for all pairs $i \in V$ and $j \in P_i$. The interpretation of z_{ij} is that $z_{ij} = 1$ if gene i is chosen to cover patient j, and $z_{ij} = 0$ otherwise. For $j \in P$, let M_j be the set of genes mutated in j. We define the following model: we keep the objective function (1) and the constraints (2), (4), (5) and add the following constraints: (7) $x_i \geq z_{ij}$, $\forall i \in V, j \in P_i$; (8) $y_j \leq w_{ij} z_{ij} + (1 - z_{ij})$, $\forall i \in V, j \in P_i$; (9) $y_j \leq \sum_{i \in M_j} z_{ij}$, $\forall j \in P$; (10) $0 \leq y_j \leq 1$, $\forall j \in P$. Note that y_j are now continuous variables and that for a feasible solution we may have that for some $j \in P : \sum_{i \in M_j} z_{ij} > 1$, however at the optimal solution $\sum_{i \in M_j} z_{ij} \leq 1$ (assuming all weights are different).

2.4 Permutation Test

We use a permutation test to assess the statistical significance of the subnetworks identified by the methods above. In particular, we generate datasets under the null hypothesis by permuting the identity of the genes in the network. The test statistic used to compute the (empirical) p-value is the value of the objective function of the solution. Given a permuted dataset and the value X of the test statistic obtained from the solution found using the real dataset, we are only interested in knowing if there is solution with test statistic $\geq X$ in a permuted dataset. Hence, we can stop an algorithm as soon as we are sure that the statistic on the current permuted dataset will be either certainly larger or certainly lower than the observed value X. This can be easily implemented in the branch and bound framework of `B&C` by adding a constraint on the value of a feasible solution, so that we can halt the search when either the lower bound of `B&C` becomes $\geq X$ or when the upper bound becomes $< X$.

We note that when a significant correlation between the degree of a node and its coverage is present, permuting the identity of the genes in the network may overestimate the significance of the subnetworks identified. We therefore checked if in our instances there was a high correlation between degree and coverage of the nodes: in all instances the absolute value of such correlation is low (≤ 0.08).

3 Results

In this section we describe the cancer data and the computing environment used in our experiments and the results obtained on the cancer datasets.

Data. We use the HIPPIE network[1] [22]. The corresponding interaction graph G consists of 15094 nodes and 188891 edges. Mutation data is obtained from the TCGA Pan-Cancer analysis[2] [18,28], with mutations of all genes measured in 3425 patients from 11 cancer types. We considered datasets of individual cancer types as well as all samples together, the latter referred to as `pancan` dataset (Table 1).

In all our experiments we performed the preprocessing described in Sect. 2, and we report running times for methods after the preprocessing.

Table 1. Cancer datasets. For each dataset, we report the number *genes* of gene nodes in graph G after preprocessing and the number $|P|$ of patients in the instance.

Dataset	Genes	$\|P\|$	Dataset	Genes	$\|P\|$	Dataset	Genes	$\|P\|$	Dataset	Genes	$\|P\|$
`pancan`	12310	3412	`coadread`	12088	495	`kirc`	11611	424	`lusc`	11752	177
`blca`	11424	100	`gbm`	11452	276	`laml`	10964	194	`ov`	11536	456
`brca`	11535	506	`hnsc`	11738	306	`luad`	11740	230	`ucec`	11865	248

Computing Environment and Solver Configuration. We implemented `B&C` in Python 2.7.5 using Gurobi 6.5.0 and callback functions. All experiments were conducted on local nodes of a computing cluster. Each node had the following configuration: two Intel E5-2680v3 CPUs with 12 CPU cores each, amounting to 24 cores in total, 64 GB RAM and 200 GB local SSD storage. All parameters in Gurobi were left at their default values, except for the number of threads that was set to one. In this way, experiments to compare the running times among different programs are conducted with serial computation. In permutation tests, we first solved the real dataset instance using a single thread and, then, the permuted datasets in parallel, one dataset per process, using python multiprocessing module and work stealing strategy.

We run the approximation and the greedy algorithms described in Sect. 2.1 and our `B&C` on the datasets of Table 1 for $k = 10$, 15, and 20. For each pair (dataset, k), Table 2 shows the coverage of the best solution found by the various algorithms and the running time (median over 10 runs). We observe that for only five of the 36 pairs (dataset, k) the approximation or greedy algorithms identify solutions with coverage as high as the the optimal found by `B&C`. Even more interestingly, the runtime of the `B&C` is comparable to the runtime of the greedy algorithm, and it is above 1 min only for 11 pairs (instance, k), and only twice above 5 min. For $k = 10$, we also compared the runtime of the `B&C` with the runtime of an ILP formulation that models connectivity constraints as in [6],

[1] http://cbdm-01.zdv.uni-mainz.de/~mschaefer/hippie/.
[2] http://compbio-research.cs.brown.edu/pancancer/hotnet2/.

Table 2. Comparison of algorithms. For each pair (dataset, k), the coverage (*cov*) of the solution reported by the various algorithms and their running time (*time* [hh:mm:ss]) are shown. In bold: coverages of solutions from `B&C` that are strictly higher then coverage of solutions from approximation and greedy algorithms; runtimes of `B&C` that are lower than runtimes of greedy algorithm.

Dataset	k	approximation		bfs		ratio		greedy		flow		B&C	
		Cov	Time	Cov	Time	Cov	Time	Cov	Time	Cov	Time	Cov	Time
pancan	10	1804	4:00:12	1804	2:08:30	-	>12 h	1469	0:00:55	1855	>18 h	**1855**	**0:00:34**
	15	2072	4:30:14	2079	2:42:24	-	>12 h	1648	0:01:48	2168	>18 h	**2168**	**0:00:38**
	20	2276	5:01:51	2277	3:10:17	-	>12 h	1817	0:02:33	2361	>18 h	**2361**	**0:02:05**
blca	10	84	2:29:19	85	1:04:08	87	6:15:03	79	0:00:34	87	9:09:23	87	0:02:06
	15	94	2:44:52	93	1:19:21	96	6:44:01	86	0:01:01	97	8:35:49	**97**	0:02:40
	20	100	2:59:28	100	1:36:49	100	7:03:14	93	0:01:29	100	4:40:14	100	0:02:23
brca	10	190	2:30:25	193	1:03:45	189	7:42:21	162	0:00:34	196	6:55:42	**196**	**0:00:09**
	15	229	2:43:46	229	1:19:40	226	8:17:40	184	0:01:03	236	6:59:14	**236**	**0:00:43**
	20	258	3:00:06	258	1:32:21	256	8:42:12	233	0:01:34	270	8:16:49	**270**	0:01:42
coadread	10	468	3:06:52	468	1:25:02	-	>12 h	454	0:00:41	472	0:14:41	**472**	**0:00:23**
	15	479	3:25:38	479	1:46:10	-	>12 h	465	0:01:13	481	6:29:51	**481**	**0:00:50**
	20	485	3:48:49	484	2:04:56	-	>12 h	473	0:01:49	488	7:31:48	**488**	**0:01:05**
gbm	10	173	2:27:19	170	1:03:22	172	6:38:31	142	0:00:32	176	1:33:48	**176**	**0:00:23**
	15	193	2:42:53	192	1:17:23	194	7:14:15	152	0:00:56	198	3:04:07	**198**	**0:00:38**
	20	209	2:58:39	209	1:34:27	210	7:39:26	158	0:01:23	215	3:11:56	**215**	**0:01:07**
hnsc	10	208	2:42:36	208	1:08:37	205	7:50:45	181	0:00:37	214	6:58:14	**214**	0:01:22
	15	241	2:56:58	241	1:23:36	240	8:36:33	206	0:01:08	248	7:01:28	**248**	**0:00:58**
	20	259	3:11:00	260	1:43:39	260	9:11:04	225	0:01:44	267	12:37:39	**267**	0:03:16
kirc	10	335	2:32:54	337	1:02:42	328	9:17:26	306	0:00:34	337	0:22:21	337	**0:00:06**
	15	352	2:48:38	353	1:19:03	350	9:49:22	321	0:01:04	359	0:32:10	**359**	**0:00:18**
	20	366	3:07:14	366	1:38:15	362	10:33:34	331	0:01:36	374	0:56:15	**374**	**0:00:34**
laml	10	96	2:07:23	97	0:55:42	93	5:10:48	79	0:00:27	98	1:56:59	**98**	**0:00:08**
	15	109	2:20:35	109	1:09:39	105	5:30:37	84	0:00:44	111	>18 h	**111**	**0:00:18**
	20	118	2:35:59	119	1:23:56	113	5:52:13	89	0:00:58	122	0:04:13	**122**	**0:00:28**
luad	10	183	2:42:34	184	1:08:58	185	9:24:50	171	0:00:36	188	4:16:21	**188**	0:00:55
	15	202	2:55:32	201	1:25:29	201	8:22:01	183	0:01:06	206	7:20:46	**206**	0:02:18
	20	213	3:12:17	213	1:41:32	213	8:58:25	189	0:01:40	219	7:25:30	**219**	0:04:55
lusc	10	159	2:43:13	160	1:10:05	159	7:36:18	145	0:00:36	160	13:24:59	160	0:04:09
	15	170	2:57:22	170	1:26:39	169	8:04:27	158	0:01:07	173	>18 h	**173**	0:14:07
	20	176	3:16:17	177	1:45:46	176	8:53:21	167	0:01:39	177	>18 h	177	0:05:38
ov	10	259	2:33:15	258	1:05:38	263	9:51:42	253	0:00:34	264	0:45:25	**264**	**0:00:14**
	15	284	2:48:59	284	1:22:26	288	10:27:14	267	0:01:00	290	2:31:51	**290**	**0:00:25**
	20	304	3:04:25	306	1:38:59	306	11:18:41	277	0:01:28	313	2:57:04	**313**	**0:00:42**
ucec	10	209	2:53:05	209	1:18:22	210	9:03:36	194	0:00:38	211	0:21:33	**211**	**0:00:12**
	15	218	3:12:04	218	1:35:53	219	9:41:43	204	0:01:07	222	9:44:56	**222**	**0:00:35**
	20	226	3:31:24	228	1:58:31	227	10:22:08	211	0:01:39	231	10:23:55	**231**	**0:00:49**

adds forbidden solution cuts whenever a nonconnected integer solution is found, and employs a min-cut flow algorithm on fractional solutions for separation (as in [6]). The results in Table 2 (column `flow`) show that the `B&C` approach we propose is much faster than this alternative ILP approach.

To test the scalability of our `B&C` algorithm, we generated one larger instance by replicating the mutations in the `pancan` dataset three times, for a total of

Table 3. Permutation test results. For each pair (instance, k) and each combination of algorithms used on real dataset and permuted datasets, the p-value (from 100 permutations) is reported.

Instance	k	p-value: real dataset/permuted datasets			Instance	k	p-value: real dataset/permuted datasets		
		B&C/B&C	bfs/B&C	bfs/bfs			B&C/B&C	bfs/B&C	bfs/bfs
pancan	10	0.01	0.02	0.01	kirc	10	0.02	0.02	0.01
	15	0.01	0.09	*		15	0.01	0.18	0.03
blca	10	0.13	0.54	0.28	laml	10	0.02	0.06	0.02
	15	0.34	0.99	0.85		15	0.02	0.08	0.02
brca	10	0.01	0.02	0.01	luad	10	0.14	0.53	0.3
	15	0.03	0.17	0.04		15	0.32	0.85	0.49
coadread	10	0.01	0.3	0.16	lusc	10	0.77	0.77	0.6
	15	0.11	0.52	0.14		15	0.69	1	0.83
gbm	10	0.13	0.55	0.39	ov	10	0.03	0.19	0.1
	15	0.18	0.75	0.39		15	0.06	0.41	0.14
hnsc	10	0.17	0.56	0.17	ucec	10	0.01	0.03	0.02
	15	0.07	0.38	0.06		15	0.02	0.36	0.12

*denotes experiments that did not complete in <2 h.

Table 4. Weighted model results. For each dataset and value of k, the weight of the optimal solution and the median runtime over 10 runs is shown.

Dataset	k	Weight	Runtime [s]	p-value	k	Weight	Runtime [s]	p-value	k	Weight	Runtime [s]	p-value
brca	10	127.06	22.29	0.01	15	137.06	141.04	0.01	20	141.98	184.16	0.03
gbm	10	93.37	130.41	0.03	15	94.26	341.02	0.02	20	94.69	479.55	0.01

10275 patients in P. On such instance, `B&C` identifies the optimal solution in 257 s for $k = 10$, 270 s for $k = 15$, and 435 s for $k = 20$.

We also compared the statistical significance of the results obtained using `B&C` for $k = 10, 15$ with the statistical significance of the results obtained using the variant `bfs` of the approximation algorithm, that reported the best solution among the approximation and greedy algorithm in most cases. In particular, we used `bfs` to obtain the best solution on the instances from Table 1 and used `bfs` in the permutation test of Sect. 2.4 to compute the p-value for such solutions. We repeated the same experiment (with the same permuted datasets) using `B&C` instead of `bfs` for both the instances in Table 1 and the permuted datasets. We also used `B&C` to compute the p-value for the solutions obtained by `bfs` on real data. Results are shown in Table 3. We observe that `B&C` almost always identifies more statistically significant solutions compared to `bfs`. Moreover, in several instances we see that the solution obtained by `bfs` appears significant when `bfs` (that does not identify the optimal solution) is used for the permuted datasets, while the significance of such a solution is greatly reduced when `B&C` (that does identify the optimal solution) is used instead.

We considered the model with weights from Sect. 2.3 and tested it on the `brca` and `gbm` datasets. Similarly to the analysis performed in [18], the weights are obtained as $-\log_{10} q_i$, where q_i is the MutSigCV[3] [17] q-value for gene i. Weight, runtime, and p-value of optimal solutions are presented in Table 4.

[3] http://firebrowse.org.

While the runtime increases, as expected for these more complicated formulation, it still remains feasible to identify statistically significant large subnetworks of high weight. For `brca` and $k = 10$, our `B&C` algorithm identifies the subnetwork containing genes {BMI1, CTCF, ELAV1L, FOXA1, GATA3, MLL3, NCOR1, PTEN, RUNX1, TBX3}. While the last 6 genes were reported as significantly mutated by single gene test in the TCGA publication on the same dataset [20], CTCF and FOXA1 are known cancer genes that did not pass significance for single gene testing in [20]. Further, the polycomb group gene BMI1 is mutated with low frequency, but has been reported to be involved in various cancers [2].

4 Conclusions and Discussion

We presented a novel algorithm for the connected maximum coverage problem, previously proposed for finding frequently mutated subnetworks in cancer. Our algorithm is based on an ILP formulation solved in a branch and cut framework. Our results show that our algorithm identifies subnetworks more frequently mutated and of higher statistical significance compared to previously proposed algorithms and to greedy approaches, while maintaining a runtime lower than or comparable to the runtime of greedy approaches. We also generalised our formulation to the case of weights for each gene in each patient, and showed that using this formulation we identify networks containing cancer genes that are not identified by single gene tests. While we considered a protein-protein interaction networks as interaction graph, our approach is also applicable when a diffusion-based influence graph [25] is used. In this work we focused on CMCP, but we believe that our framework could be beneficial to other optimization problems in bioinformatics where connected subgraphs are sought.

Acknowledgments. Computation for the work described in this paper was supported by the DeiC National HPC Center, University of Southern Denmark. This work is supported, in part, by MIUR of Italy under project AMANDA and by NSF grant IIS-1247581, and has been done, in part, while FV was a research fellow at the Simons Institute for the Theory of Computing (University of California, Berkeley). The results presented in this manuscript are in whole or part based upon data generated by the TCGA Research Network: http://cancergenome.nih.gov/.

References

1. Álvarez-Miranda, E., Ljubić, I., Mutzel, P.: The maximum weight connected subgraph problem. In: Jünger, M., Reinelt, G. (eds.) Facets of Combinatorial Optimization, pp. 245–270. Springer, Heidelberg (2013)
2. Benetatos, L., Vartholomatos, G., Hatzimichael, E.: Polycomb group proteins and MYC: the cancer connection. Cell. Mol. Life Sci. **71**(2), 257–269 (2014)
3. Bruckner, S., Hüffner, F., Karp, R.M., et al.: Topology-free querying of protein interaction networks. J. Comput. Biol. **17**(3), 237–252 (2010)
4. Creixell, P., Reimand, J., Haider, S., et al.: Pathway and network analysis of cancer genomes. Nat. Methods **12**(7), 615–621 (2015)

5. Ding, L., Wendl, M.C., McMichael, J.F., et al.: Expanding the computational toolbox for mining cancer genomes. Nat. Rev. Genet. **15**(8), 556–570 (2014)
6. El-Kebir, M., Klau, G.W.: Solving the maximum-weight connected subgraph problem to optimality. CoRR abs/1409.5308 (2014)
7. Vandin, F., Upfal, E., Raphael, B.J.: Algorithms and genome sequencing: identifying driver pathways in cancer. Computer **45**(3), 39–46 (2012)
8. Fischetti, M., Leitner, M., Ljubic, I., et al.: Thinning out steiner trees: a node based model for uniform edge costs. Math. Progr. Comput. (2015, submitted)
9. Garraway, L.A., Lander, E.S.: Lessons from the cancer genome. Cell **153**(1), 17–37 (2013)
10. Hanahan, D., Weinberg, R.A.: Hallmarks of cancer: the next generation. Cell **144**(5), 646–674 (2011)
11. Hochbaum, D.S.: Approximation Algorithms for NP-hard Problems. PWS Publishing Co., Boston (1996)
12. Hudson, T.J., Anderson, W., Aretz, A., et al.: International network of cancer genome projects. Nature **464**(7291), 993–998 (2010)
13. Johnson, D.S., Koch, T., Werneck, R.F., et al.: The eleventh dimacs implementation challenge. http://dimacs11.cs.princeton.edu/home.html
14. Kim, Y.A., Cho, D.Y., Dao, P., et al.: Memcover: integrated analysis of mutual exclusivity and functional network reveals dysregulated pathways across multiple cancer types. Bioinformatics **31**(12), i284–i292 (2015)
15. Kim, Y.A., Salari, R., Wuchty, S., et al.: Module cover-a new approach to genotype-phenotype studies. Pac. Symp. Biocomput. 135–146 (2013)
16. Kuo, T.W., Lin, K.C.J., Tsai, M.J.: Maximizing submodular set function with connectivity constraint: theory and application to networks. IEEE/ACM Trans. Netw. **23**(2), 533–546 (2015)
17. Lawrence, M.S., Stojanov, P., Polak, P., et al.: Mutational heterogeneity in cancer and the search for new cancer-associated genes. Nature **499**(7457), 214–218 (2013)
18. Leiserson, M.D., Vandin, F., Wu, H.T., et al.: Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes. Nat. Genet. **47**(2), 106–114 (2015)
19. Ljubić, I., Weiskircher, R., Pferschy, U., et al.: An algorithmic framework for the exact solution of the prize-collecting steiner tree problem. Math. Program. **105**(2–3), 427–449 (2006)
20. TCGA Network: Comprehensive molecular portraits of human breast tumours. Nature **490**(7418), 61–70 (2012)
21. Raphael, B.J., Dobson, J.R., Oesper, L., et al.: Identifying driver mutations in sequenced cancer genomes: computational approaches to enable precision medicine. Genome Med. **6**(1), 5 (2014)
22. Schaefer, M.H., Fontaine, J.F., Vinayagam, A., et al.: Hippie: integrating protein interaction networks with experiment based quality scores. PLoS One **7**(2), e31826 (2012)
23. Shrestha, R., et al.: HIT'nDRIVE: multi-driver gene prioritization based on hitting time. In: Sharan, R. (ed.) RECOMB 2014. LNCS, vol. 8394, pp. 293–306. Springer, Heidelberg (2014)
24. Ulitsky, I., Karp, R.M., Shamir, R.: Detecting disease-specific dysregulated pathways via analysis of clinical expression profiles. In: Vingron, M., Wong, L. (eds.) RECOMB 2008. LNCS (LNBI), vol. 4955, pp. 347–359. Springer, Heidelberg (2008)
25. Vandin, F., Upfal, E., Raphael, B.J.: Algorithms for detecting significantly mutated pathways in cancer. J. Comput. Biol. **18**(3), 507–522 (2011)

26. Vogelstein, B., Kinzler, K.W.: Cancer genes and the pathways they control. Nat. Med. **10**(8), 789–799 (2004)
27. Vogelstein, B., Papadopoulos, N., Velculescu, V.E.: Cancer genome landscapes. Science **339**(6127), 1546–1558 (2013)
28. Weinstein, J.N., Collisson, E.A., Mills, G.B., et al.: The cancer genome atlas pan-cancer analysis project. Nat. Genet. **45**(10), 1113–1120 (2013)
29. Wang, Y., Buchanan, A., Butenko, S.: On imposing connectivity constraints in integer programs (2015). http://www.optimization-online.org/DB_HTML/2015/02/4768.html