Survey on Multi-Commodity Flow Problems

Henry Li Xinyuan

Spring 2021

Contents

1 Introduction

Multi-commodity flow problems are natural generalisations of the $s-t$ flow problem. Given a graph $G=(V,E)$ where $|V|=n$ and $|E|=m$, instead of having a single source-sink pair (s,t), we are asked to route flows between k such pairs $C_i=(s_i,t_i), i\in[1..k]$.

Numerous subproblems exists within this general framework, differing from each other by the addition/removal of one or more constraints. While many of these subproblems allow the use of similar techniques, these subproblems exhibit very different levels of hardness in both calculation and approximation. In addition, the graph-theoretic properties of G such as directness, being a tree, treewidth, and many others, can have a major effect on the difficulty of this problem.

1.1 Formal definition: general case

- Input: Acyclic graph $G=(V,E)$, k commodities $C_i=(s_i,t_i,d_i), i\in[1..k]$

  Intuitively, d_i denotes the total amount of flow that gets routed through the graph. It is not fixed in the max-flow subproblem, where it is instead the objective to be maximised. Problems with $d_i=1\forall i\in[1..k]$ are often much easier both to solve exactly and to approximate.

- Feasible solutions: Let $\mathcal{P}_i, i\in[1..k]$ denote the set of all paths from s_i to t_i, a feasible solution would be the set f_{P_i} for each $P_i\in\mathcal{P}_i, i\in[1..k]$.

  f_{P_i} should take values between 0 and 1, indicating the proportion of the total flow between s_i and t_i that uses P_i. If the flow of a single commodity could be split between multiple paths, then $f_{P_i}\in[0,1]$; otherwise, $f_{P_i}\in 0,1$. In either case, $\sum_{P_i\in\mathcal{P}_i} f_{P_i}=1$ for all $i\in[1..k]$.

1.2 Integrality

As is the case with single-source max-flow, multi-commodity flow has LP-formulations that are considered solvable in polynomial time. The Ford-Fulkerson algorithm/method, being commonly known as a poly-time solution for single-source max-flow, can also be employed in the case of multi-commodity max-flow by adding a ”common source” and a ”common sink”, as well as a path (with unbounded capacity) from the common source to each source as well as from each sink to the common sink [17]. On the other hand, unlike in the case of single-source max flow where an integral solution can be found in polynomial time, in general the integral multi-commodity flow problem is NP-complete. In fact, many subproblems have been shown to be hard to approximate within a constant factor, especially if we limit ourselves to common methods such as LP rounding.

Even in the non-integral case, there is substantial interest in approximating multi-commodity flow problems, mostly due to its wide range of applications and the need for improved performance for many of its applications. Many of these approximation methods were then appended with a solution rounding scheme to produce betther techniques for the integral version of the problem.

On the other hand, integral multi-commodity flow has a slightly simpler formulation than its non-integral counterpart (in the case where flow can be broken down into individual parts):

- Input: Graph $G=(V,E)$, k commodities $C_i=(s_i,t_i), i\in[1..k]$
- Feasible solutions: Paths $P_i, i\in[1..k]$ such that $P_i\in\mathcal{P}_i$, where $\mathcal{P}_i, i\in[1..k]$ denote the set of all paths from s_i to t_i.

1.3 Directedness

The formulation for the directed version of the problem is simple: we add a new input $dir(e), e=(p_1,p_2)\in E$ which takes values 0 or 1. The paths using e must then use e in the forward or backward direction only depending on the value of $dir(e)$. Later on, we will look at the directed version of this problem in more depth as a special graph property.

1.4 Related problems

- Cut problems

  Cut problems are, as the name suggests, problems that look for a subset of E or V which satisfy certain requirements, usually that are separating one or multiple source-sink pairs. As we will discuss in the next section, cut problems are inherently related to flow problems due to the dual relationship between their LP formulation. THe flow-cut gap is usually equivalent to the integrality gap of the flow problem, which makes these two problems basically interchangeable.

- Packing problems

  The packing problem involves choosing a set of items with specified space requirements, to be placed in a set of containers with limited size. Packing problems can be seen as a generalisation of the multi-commodity flow problem: each edge or each path could be thought of as a container and each flow could be thought of as an item. As is the case with flow problems, integral and fractional packing problems have rather distinct flavours. There have been works that attempt to address multi-commodity flow from the angle of packing problems [18, 22, 51].

- Covering problems

  Those are the duals of packing problems.

- Matching problems

  If we add node-disjointness as a constraint, certain special cases of multi-commodity flow problems can be reduced to a matching problem [29]. Indeed, we see that used as an important tool when the input graph is a grid or can be converted to a grid [5, 6, 7].

2 LP formulation and Dual

We look at the LP formulation of the non-integral max-throughput subproblem. We change the formulation slightly, defining $g_{P_i} = f_{P_i} \cdot d_i$ in order to keep all constraints linear. Recall that in the case of single-commodity max-flow, the dual LP is exactly the formulation for the min-cut problem, leading to the max-flow min-cut theorem which states that the solutions to these two problems are equivalent.

$$
\begin{aligned}
&\text{Maximise} \sum_{i\in[1..k]} \sum_{P_i\in\mathcal{P}_i} g_{P_i} && && \text{Total flow}\\
&g_{P_i} \geq 0 && \forall i \in [1..k], \forall P_i \in \mathcal{P}_i && \text{Flows are at least 0}\\
&\sum_{P_i : e\in P_i} g_{P_i} \leq c(e) && \forall e \in E && \text{Flow does not exceed capacity}
\end{aligned}
$$

We observe that the number of LP equations in this instance is $O(|\sum_{i\in[1..k]} \mathcal{P}_i|)$, which is exponential. We would therefore need to convert into a compact formulation in order to solve the LP in polynomial time.

We next look at the dual of the above LP:

$$
\begin{aligned}
&\text{Minimise} \sum_{e\in E} c(e) \cdot y(e)\\
&y(e) \geq 0 && \forall e \in E\\
&\sum_{e\in P_i} y(e) \geq 1 && \forall i \in [1..k], \forall P_i \in \mathcal{P}_i
\end{aligned}
$$

We observe that if we limit the values of $y(e)$ to be integral (in $0, 1$), and interpret each as whether we include edge e in a cut, we have ourselves an LP formulation for the min-multi-cut problem. The objective becomes minising the total cost of a cut, the positivity constraint becomes the integral constraint, and the last constraint requires that the cut separates each source from its corresponding sink.

Finally, due to the minor tweaks require to adapt the above natural LP formulation to the various subproblems of multi-commodity flow (some of which we don't have a good understanding of yet), they tend to have differing integrality gaps which will be explored individually for each subproblem.

2.1 Compact LP formulation

Note that the above LP formulation, while intuitive, has an exponential number of constraints. Garg, Vazirani and Yannakakis proposed a more compact formulation, defining $flow_e^i, e \in E, i \in [1..k]$ as the amount of flow of commodity i passing through edge e [2]. An edge is then added between (t_i, s_i) for each $i \in [1..k]$ with unbounded capacity, where only the flow of commodity i is allowed, so that each flow becomes cyclical and we get a single edge where the total flow of a commodity can be measured:

$$\begin{array}{llr} \text{Maximise} \sum_{i\in[1..k]} flow^i_{(t_i,s_i)} & & \text{Total flow} \\ flow_e^i \geq 0 & \forall i \in [1..k], \forall e \in E & \text{Flows are at least 0} \\ \sum_{i\in[1..k]} flow_e^i \leq c(e) & \forall e \in E & \text{Flow does not exceed capacity} \\ \sum_{e\in E: e=(v,v_1)} flow_e^i = 0 & \forall v \in V, \forall i \in [1..k] & \text{Flow in and flow out should be equal} \end{array}$$

2.2 Converting a solution to max multi-flow into a solution to min multi-cut

As is the case with the single-commodity max-flow problem, a solution to the max multi-flow problem can be efficiently converted into a solution to min multi-cut by using the residual graph of the max flow.

First, subtract from the capacity $c(e)$ for each edge, the total amount of flow using the edge $\sum_{P_i : e \in P_i} g_{P_i}$. The result is the residual graph of the flow solution.

Next, we trace out all of the vertices that are reachable from each source node in the residual graph. We would end up with $\leq k$ many connected components. The min multi-cut solution are the boundary edges between these components, as well as between the components and the remaining vertices in the graph. This process also reveals the intuition that the min-cut solution corresponds to the bottleneck of the max-flow.

2.3 Primal-dual interior point algorithms in multi-commodity flow

The primal-dual algorithm is a method for solving linear programs that might be faster than other popular methods, such as the simplex method, in the context of network design problems [30]. A key idea in the primal-dual method is complementary slackness: we consider a solution set x as satisfying the *primal complementary conditions* if for its i^{th} variable x_i and the corresponding i^{th} constraint $A^T y \leq c$, $x_i > 0 \implies A^i y = c_i$. The *dual complementary conditions* is the same thing the other way round. The classic primal-dual method for calculating the exact value of an LP involves setting an initial feasible value to the dual solution, then proceed to move in the direction with better dual objective function value until a priaml solution satisfying complementary slackness is found (at which point the primal solution is optimal).

[30] proposed relaxing the complementary slackness conditions so that the primal-dual method can be used in an integral approximation context, where the optimal values don't necessarily satisfy the original complementary slackness conditions. The primal complementary slackness is still enforced (since we don't care about the integrality of the dual) but the dual complementary slackness is relaxed to take into account the integrality constraints.

[29] used the approach in [30] for the multi-commodity flow problem in trees, relaxing the dual complementary slackness condition of the "full-separation requirement" (In the dual problem of multicut: the total distance of the path between every source-sink pair is exactly 1) to an "over-separation requirement" (the total distance of the path between every source-sink pair is between 1 and 2). They were thus able to achieve an integrality gap of 2.

2.3.1 Column generation

Column generation is a technique used to solve large linear programs. It relies on the idea that the vast majority of variables in an LP doesn't have an effect on the objective function; as such, a subproblem consisting of the variables that will have an effect on the objective function (the columns corresponding to those variables in the LP matrix) are selected recursively and solved. The connection between column generation and primal-dual stems from the use of the dual solution to evaluate how well one solution to subproblem fits in the context of the whole problem, in such a way that could be reminiscent of a relaxed version of the complementary slackness conditions [46]. In addition, it is possible to combine the original primal-dual method with the column generation method : the suboptimal solutions from the early interations of a primal-dual run are often more suitable as the initial values for column generation than those from the early iterations of a simplex run [47].

3 Subproblems

Here we discuss a number of common subproblems of multi-commodity flow, in varying detail according to their presence in recent literature.

3.1 Minimise congestion (load balancing)

In the load balancing problem, the objective is to minimise the maximum number of paths that use each edge in G. If we introduce a bandwith $b(e)$ for each edge, let congestion $cong(e) = \frac{\sum_{i\in[1..k]} f_i(e)\cdot d_i}{b(e)}$, the objective would be to minimise $cong(e)$. A special case would be if $b(e) = 1$ for all $e \in E$. Another type of load is known as residual load, given by $c(e) - \sum_{i\in[1..k]} f_i(e) \cdot d_i$, that is to be maximised. Other load functions that are non-linear are commonly used as well and can render the common LP formulation of the problem problematic. An example is the Kleinrock load, defined as $\frac{\sum_i flow_e^i}{c(e)-\sum_i flow_e^i}$. [43] shows that methods for concurrent flow can be employed in the case of load balancing by treating residual edge load as commodities that require routing by converting load into a load vector; this technique works even on non-linear load functions.

This is probably the subproblem that garners the greatest amount of interest from those using multi-commodity flow in applied scenarios, such as network design. For this reason, this problem is very well-studied in the dynamic case.

3.2 Max-throughput with edge capacity

As the name of the problem suggests, in this scenario d_i is not fixed, and is instead used to represent the amount of flow that is routed for commodity i. An additional constraint, edge capacity, is in place as the main constraint, representing the maximum amount of flow that could be routed through each edge: $c_e, e \in E$. We define $cong(e)$ as we did for load balancing (with bandwith 1), and we require that $cong(e) \leq c(e)$ for all $e \in E$. We then seek to maximise $\sum_{i\in[1..k]} d_i$.

This problem is the multi-commodity counterpart to the original max-flow problem, and thus is one of the most well-studied subproblems in multi-commodity flow.

When it comes to the edge capacities, Chekuri, Khanna and Shepherd pointed out in [5] that when $c_e = b$ for some constant b, the optimal solution as given by the LP solution grows by a factor of b compared to when $c_e = 1$ (for fairly straightforward reasons). Multi-commodity flow problems where the input graph has this property are known as constant-congestion flow problems, and are relatively well-studied.

3.3 Minimise cost

In this problem, we define a cost $a(e), e \in E$ for sending a unit of flow along a path. With the demand for each commodity fixed in the input, the objective is to minimise $\sum_{e\in E} a(e) \cdot \sum_{i\in[1..k]} f_i(e) \cdot d_i$. Unfortunately, unlike the max-flow problem, the dual min-cost multi-commodity flow isn't particularly clean or well-studied. This problem can be seen as a special case of the concurrent flow problem discussed below.

3.4 Concurrent flow (max-min fairness)

The intuition behind concurrent flow is to maximise the minimum amount of flow routed for each commodity (or the proportion of demanded flow of each commodity). In this instance, we introduce a variable a indicating the proportion of desired flow each commodity is sending. We add a constraint $\sum_{P_i \in \mathcal{P}_i} f_{P_i} \geq a$, $\forall i \in [1..k]$. We then maximise a.

While techniques based on LP rounding can obviously be employed for concurrent flow, its approximation ratio cannot exceed the integrality gap $On^{1/(c+1)}$, where c denotes the congestion allowed on the edges. [52] was an early work which broke away from LP-rounding based methods, using a method which involves starting with an initial, infeasible flow and iteratively re-routing the flow. This key idea was later used in various other works in various subproblems of multi-commodity flow.

The dual of the LP formulation for concurrent flow is known as the sparcest-cut problem, which is more often how this problem is known as. Of particular interest is the wide applications of the sparcest-cut problem, especially as subroutines for other techniques such as divide-and-conquer [9]. The flow-cut bound of sparcest cut has be found to be $\Omega(logk)$ [14]. Arora, Rao and Vazirani, in the seminal work to date, used a technique grounded in semi-definite programming (SDP) which improved the approximation ratio of the problem to $O(\sqrt{\log n})$ [14].

The algorithm in [14] works by mapping the vertices of a graph onto a unit circle in $\mathbb{R}^n$, such that the average squared distance between the vertices is a fixed constant ρ. A value for such a mapping is defined as the sum of squared distances between the endpoints of a subset of edges in G. A mapping of minimum value, corresponding to the desired min-cut, could be found in polynomial time using semi-definite programming. The SDP formulation is given as follows:

$$\text{Minimise} \sum_{e_0, e_1 \in E} |v_{e_0} - v_{e_1}|^2$$

$$|v_i - v_j|^2 + |v_j - v_k|^2 \geq |v_i - v_k|^2 \quad \forall i, j, k \qquad \text{Triangle inequality}$$

$$\sum_{i<j} |v_i - v_j|^2 = 1 \quad \forall i, j, k \qquad \text{Mapping on unit sphere}$$

[14] then proceeds show that the SDP has an integrality gap of $O(\sqrt{\log n})$.

Subsequent improvements were only made on specific graph types: [54] suggested a 2-approximation that runs in $|V|^{O(tw(G))}$ time on treewidth-bound graphs, with $tw(G)$ being the max treewidth of the input graph; [55] proposes a $(1+\epsilon)$-approximation on a family of graphs that are not "small set expanders" - graphs with small subsets that have a large number of edges compared to its size - that follows the same general principle as [14].

3.5 Edge-disjoint paths and node-disjoint paths

These two problems can both be thought of as instances of integral multi-commodity flow problems with extra constriants, in particular as generalisations of the constant-congestion max-flow problem. The edge-disjoint graph problem requires that each edge only participate in one flow, and can thus be thought of as a unit-demand multi-commodity unsplittable flow problem. The node-disjoint graph problem requires that each node be part of no more than one flow. It has been shown that the NDP problem is more general than the EDP problem: one can easily convert an EDP problem into an NDP problem by breaking edges into pairs and adding additional vertices in between these pairs.

3.6 Largest satisfiable commodity subset

Also called "all-or-nothing" multi-commodity flow, this problem concerns finding the largest possible subset $S \subseteq \{C\}$ such that S is satisfiable in G. Its integral dual formulation corresponds to the k-Multicut problem. This problem is shown to have a polylog integrality gap on general graphs [57]. Most recently, Zhang, Zhu, and Luan have achieved an $O(\sqrt{q \log n})$ approximation ratio for the k-Multicut on general undirected graphs, as well as a $O(\sqrt{q})$ ratio for trees [3].

3.7 Flow over time

Also known as dynamic flow problems, these problems were introduced by Ford, Fulkerson [38] back in 1958. It has been shown to be NP-hard even in the single commodity case [41].

This family of problems have not seen as much theoretic analysis as the static multi-flow problem. On the other hand, these problems have important applications in areas where static flow problems are insufficient, thus a lot of experimental analysis has been performed on the various techniques in calculating or approximating dyanmic flow. A lot of interesting issues emerge when time comes into the equation, such as the storage of flow in intermediate nodes, the associated capacity and cost of storage, tansit time over edges, taking advantage of loops (which is never useful in the static case), etc. Fleischer and Skutella showed that storage in intermediate nodes is not beneficial in the single-commodity case [42], although the same is not true for the multi-commodity problem.

Recent works on this include [39] and [40], both of which deal with the min-cost variation. [39] is an experimental study on the problem where intermediate node storage is allowed: the flow decomposition method (decomposing flows into paths) is employed and allowed for a comparable or better experimental performance than dynamic LP solving. [40] proposed an exact algorithm to the same problem, assuming no intermediate node storage, using a non-compact LP-based model and a column generation approach. Neither of these works provided a theoretical performace or complexity to their method, which is something that could be explored further.

3.8 Flow with incomplete information

While this problem and its solutions are hard to quantify in theory, in real life there are many situations which call fo devising multi-commodity routing methods that could deal with a certain amount of missing data (usualy part of the graph or a subset of demand values) and come up with solutions that are reasonably close to being feasible. [53] is one recent study of this kind on concurrent flows, using traditional LP rounding methods and giving an experimental analysis.

3.9 Fractional flow: exact calculation and approximation

As we mentioned earlier, the fractional multi-commodity flow problem could be solved in polynomial time, by finding the solutions to the compact linear program formulation. Nevertheless, faster calculation, or $(1 + \epsilon)$-approximation, of the fractional answer is called for due to that part often being the one that dominates the total time complexity of the integral approximation theorem. One of the earliest polynomial-time approximation schemes was developed by Shahrokhi and Matula in 1990 [36]. The number of works on this topic have exploded since.

Note: we initially envisioned comparing the fractional flow approximation algorithms and combining them with appropriate rounding schemes, in an effort to benchmark the real time complexities of many of the integral multi-flow approximation techniques. It turns out that the vast majority of those techniques (with the exception of combinatorial ones) employ some form of ”fracional-rounding” process anyway; using an approximation scheme for the fractional answer is therefore often a separate problem akin to trading result precision for running time.

Approximation of fractional multi-commodity flow tends to focus on two different parameters: ϵ (or rather, ϵ^{-1}) and k (which is often assumed to be much greater than $|V|$). There tends to be a significant difference between the approaches that emphasise optimising either.

It is impossible to introduce fractional flow approximation without mentioning Lagrangian relaxation. Lagrangian relaxation is a method of linear-program decomposition, which relaxes the original problem by dividing the constraints into 2 or more parts. In the context of multi-commodity flow, this usually involves breaking down the original graph into a set of trees rooted at sources and with sinks as their leaves [35]. The vast majority of approximation schemes for fractional multi-commodity flow employ Lagrangian relaxation. It is worth mentioning that approximation schemes based on Lagrangian relaxation appear to have a guarateed quadratic dependence on ϵ^{-1} [44].

On the other side of the spectrum, Nesterov devised in 2004 an method for optimising non-smooth convex functions which can naturally be applied to linear programs and thus to the multi-commodity flow problem [45]. This method was later improved on to give an $O(\text{poly}\log(\frac{1}{\epsilon}\log_2 1/\epsilon))$-approximations for max multi-flow [48], as well as an $O(\text{poly}\log(k^2|E|^2 1/\epsilon))$-approximation for the same problem [49].

In practice, these approaches vary in performance given the problem. [58] gave an experimental analysis which suggests that column generation might be faster in smaller problems while Lagrangian relaxation excel at large-scale networks with large number of commodities.

3.9.1 Fractional max multi-flow

As a basic benchmark, we refer to the simplex algorithm with a polynomial-time average time complexity. [31] suggested a combinatorial algorithm which performs a $(1+\epsilon)$-approximation of the fractional problem in $O(\epsilon^{-1}k|E|^2|V|^2)$. The same idea was improved on in [32], improving the time complexity to $O(\epsilon^{-1}k|E|^2\log|V|)$. Both of these ideas are based on iteratively calculating the shortest path, routing flow through those paths, and increasing the length of said path. Common pathfinding algorithms are central to these techniques; [33] improved on [32] by using dynamic shortest path algorithms, which resulted in better actual runtime (without improving the theoretical runtime). [34] tweaked the approach in [32] and [33] by changing the amount of flow sent in each iteration: it sends a large amount of flow at the start, then gradually decreases the amount in proportion to the shortest path length. A run-time improvement is reported by [34] although no theoretical improvement is found.

The result in [31] remains the best result when the dependence of running time on ϵ is limited to ϵ^{-1}, whereas subsequent works on [32] have improved the theoretical time complexity to $O(|E||V|\epsilon^{-2})$ [37]. This improvement is achieved, once again, by fine-tuning the shortest-path calculation: the author of [37] observed that many of these calculations overlap with each other, and revises previous works to maintain a table of the approximate shortest paths in the graph.

3.9.2 Fractional concurrent flow

The key approximation steps for other subproblems are very similar as those for max multi-flow, thus related but slightly different results have been achieved. Karakostas proposed a $O(\text{poly}\log(|E|^2+k|V|)\epsilon^{-2})$-approximation based on Lagrangian relaxation (assuming all the values in the input are bounded by a constant; if that's not true, the time complexity would be $O(\text{poly}\log|E|^2\log M + k|V|)$ where M is the largest number in the input and $logM$ is its representation in binary). In particular, it is argued in [35] that if we don't need the approximate flow of each commodity in each edge, the runtime could be further reduced to $O(\text{poly}\log|E|^2\epsilon^{-2})$. More recently, Mądry improved that result to $O(\text{poly}\log(|E|+k)|V|\epsilon^{-2}\log(M))$ using a very similar method [37].

4 Special graph types

As the graph is central to any flow problem, special properties of the graph can have huge effects on the difficulty of calculating or approximating the multi-commodity flow problem, integral or not. Generally speaking, extra constraints on the graph can reduce the approximation ratio of of both integral and fractional multi-commodity flow; we will be focusing on these instances here. On the other hand, examples have been constructed for certain special graphs to show that no graph-specific improvements exist for certain subproblems.

4.1 Directed graphs

While the techniques are mostly similar, directed multi-commodity flow problems sometimes have lower integrality gaps than their undirected counterparts.

In the context of the integral multi min-cut problem, note that a trivial k-approximation exists [9]:

1. For each commodity $C_i, i \in [1..k]$, calculate its min-cut

2. Return the union of the min-cuts from step 1.

The approximation ratio is k since the total cost of the union is at most k times the highest-cost min-cut among the k commodities, which in turn is less than or equal to OPT. Subsequent results from Saks, Samorodnitsky and Zosin showed that the integrality gap is no less than $L - \epsilon$ for any $\epsilon > 0$ [10].

Since the value of k is independent of the graph itself, the question of minimum integrality gap in terms of $|V|$ or $|E|$ remains open: a lower bound of such a gap has been given as $\Omega(\log|V|/\operatorname{poly}\log|V|)$ [20], where $\operatorname{poly}\log|V|$ denotes $\Theta((\log|V|)^c)$ for some constant c. The current best result is $O(|V|^{11/23} \cdot \operatorname{poly}\log|V|)$ [19]. Not a lot of recent work has managed to improve on this result. On the other hand, this problem has been shown to be hard to approximate to within a factor of $2^{\Omega(log^{1-\epsilon}|V|)}$ for any constant ϵ, assuming that some problems in NP don't have efficient randomised algorithms [20].

In the case of concurrent flow (sparcest cut) in a directed graph, the lower bound on the approximation ratio has similarly been shown to be at least $\Omega(\log|V|/\operatorname{poly}\log|V|)$ [20]. Finally, in the case of the node-disjoint paths problem, the current best lower bound on the approximation ratio is given at $O(|E|^{1/2-\epsilon})$ [56].

4.2 Trees

A tree is a very special case of this problem, since a unique path exists between each two nodes in a tree. It is probably the graph type that is most studied in the context of multi-commodity flow, aside from general case graphs. In the LP formulation we previously gave, the number of positivity constraints dropped from exponential of $|V|$ to linear on k, allowing for the fractional case to be solved by LP in polynomial time.

When all demands are set to 1, Garg, Vazirani and Yannakakis showed in 1997 that the integral gap, and thus the flow-cut gap, for the multi-commodity flow problem in a tree is 2 [29]. When adding arbitrary demand into play, Chekuri, Mydlarz and Shepard [1] remarked that the above mentioned LP formulation has a constant factor integrality gap, as long as the demand for each item is bounded by a constant or some factor polynomial to the other constraints (graph size, edge capacities, etc.). Specifically, a 4-approximation was found for the unit-demand case, whereas a 48-approximation was proposed for a tree where $\max(d_i) \leq \min(c_e)$.

Certain special instances of trees allow for the some versions of the problem to be solved in polynomial time. Dressler and Strehler pointed out one such example for max multi-flow: if the number of distinct sources or sinks is bounded by a constant and if the input graph is a tree, then the problem can be solved in polynomial time [16]. If the tree is what is known as a "spider" - a tree with only one vertex of degree greater than 2 - then a polynomial-time exact algorithm exists as well [51].

Interestingly, if we relax the edge capacity constraints to allow for some additive overhead, it is possible to find algorithms that match the optimal solution in the original problem in polynomial time using an iterative LP relaxation method [51].

4.2.1 Star

By removing the centre v of the star, and replacing each edge $e = (v, v_i)$ that is part of the path between a source-sink pair with a new node v_e and a new edge (v_i, v_e), the problem can be converted into a max matching problem (or a max c-matching problem, if the graph has capacity greater than 1) [1]. As such, it can be calculated in polynomial time with any algorithm for max matching.

4.2.2 General graph with bound on treewidth

While not a graph category on its own, the treewidth of a graph can have substantial effects on the techniques used in multi-commodity flow problems. Informally, treewidth represents the how far a graph is from being a tree: it measures the minimum width of all possible tree decompositions of a graph. A tree would have a treewidth of 1, where as complete graph of size n would have a treewidth of $n - 1$. Note that calculating the treewidth of a graph is NP-hard (as is getting the tree decomposition corresponding to the treewidth) [21], thus calculating a treewidth in order to choose an appropriate algorithm is not possible.

As we have discussed previously, trees have special properties that allow for much lower approximation ratios for a variety of multi-commodity flow problems. Thus, if the tree decomposition of a graph is provided, we can take advantage and reduce the problem into a set of subproblems involving the trees in the decomposition. [15] is a recent work on the edge-disjoint paths problem in the context of treewidth-bounded graphs. Let r be the treewidth of a graph, [15] shows that the integrality gap for the standard LP relaxation of multi-commodity flow is $O(r^3)$.

4.2.3 General graph with tree-likeness

Here tree-likeness is referred to as the minimum number of vertex deletions that would transform a graph into a tree. A recent work has found that graphs with low vertex deletion distance r from a tree can have better approximation ratios for EDP: $O(\sqrt{r}\log kr)$ as opposed to $O(\sqrt{|V|})$ on general graphs.

4.3 Planar graphs

Roughly speaking, a planar graph is a graph which does not have edges intersecting (aside from in vertices) when embedded into a plane. In order to explore this special case, we need a few definitions:

Definition 4.1 *For a subset $A \subseteq V$ of vertices in G, $\delta(A)$ refers to the set of edges in E such that exactly one endpoint is in A.*

Definition 4.2 *A set T of terminals is well-linked if for each subset $S \subseteq V$ with $|S\hat{X}| \leq |(V\ S)\hat{X}|$, $|\delta(S)| \geq |S\hat{X}|$.*

Definition 4.3 *A graph H is a minor of G if it can be formed from G by deleting edges and vertices and by contracint edges.*

The interesting aspect of planar graphs for multicommodity flow is the following result, due to Robertson, Seymour and Thomas [7]:

Theorem 4.4 *In a planar graph, a well-linked set of vertices $|T|$ implies the existence of a grid minor of size $\Omega(|T|)$.*

Here a grid minor refers to a minor that is also a grid. An alternative way of stating this theorem is that, for every grid H, every graph whose treewidth is large enough with respect to $|V(H)|$ contains H as a minor [12].

In other words, it is possible, with some overhead, to reduce a planar-graph into a grid, where it is much eaiser to calculate/approximate certain types of max multi-flow than in general graphs. For example, there are special techniques which can be applied to the edge-disjoin path problem in planar graphs with edge capacity at least 2, which allow for the integrality gap to drop from $\Omega(\sqrt{n})$ to a constant [26].

The relationship between the treewidth of the original graph $tw(G)$ and the edge length of the associated grid len is an area of active research: [12] improved the upper bound of len from $len = \Omega(\sqrt{\log tw(G)/\log\log tw(G)})$ to $O(\sqrt{tw(G)/\log tw(G)})$, which in turn helps with reducing the overhead of converting the grid solution back into the planar graph solution.

Kawarabayashi and Kobayashi showed that the all-or-nothing (unsplittable) multi-commodity flow problem has a constant factor (8) integrality gap when the input graph is planar [6]. However, the subproblem with the greatest tendency to focus on planar graphs are EDP and NDP. Okamura and Seymour produced the following result early on which laid the foundation for related later works:

Definition 4.5 *For some subset $S \subseteq V$, a demand is supported on S if there is no demand into or away from S.*

Definition 4.6 *In a planar graph, a face of a graph is a set of vertices and edges bounding a region, with no vertex or edge enclosed within.*

Theorem 4.7 *Okamura-Seymour: In a planar graph G, if the demand is satisfied on every face of G, then there is no flow-cut gap. [59]*

This result works on all edge-capacitated planar graphs; [60] showed that will it does not apply to node-capacitated planar graphs, it is possible to convert it into an approximate result with a ϵ-fraction instead, thus allowing for EDP and NDP results on planar graphs to be connected.

More recently, Séguin-Charbonneau and Shepherd, building on previous work by Chekuri, Khanna and Shepherd [27] which gave the same result for congestion 4, gave a constant factor approximation algorithm for the edge-disjoint path problem in planar graphs with congestion $c(e) = 2 \forall e \in E$ [25]. By contrast, the best approximation ratios for congestion 2 EDP on general graphs is at $O(\text{poly} \log k)$ [28]. Unfortunately, general-case EDP on planar graphs has no better known approximation ratio than $O(\sqrt{n})$, the LP integrality gap on general graphs.

4.3.1 Grids

A grid is a special case of the planar graph; a lot of applications of multi-commodity flow problems also concern grid-shaped input graphs. The multi commodity flow problem was not very well studied on grids until very recently, when Chuzhoy and Kim improved the approximation ratio of the node-disjoint paths problem from $\log n$ (same as for general graphs) to $O(n^{1/4} \cdot \log n)$ [11].

The algorithm works by distinguishing between two types of commodities, "good" and "bad", where bad commodities are those which are close to the grid boundary and good ones being the rest. Let $\Gamma(G_0)$ be the boundary of any grid G_0. Formally, the line between good and bad is drawn when $d_\infty(s_i, \Gamma(G)) \le 4\sqrt{N} + 4$ and $d_\infty(t_i, \Gamma(G)) \le 4\sqrt{N} + 4$. On a $(N \times N)$ grid the routing of "good" commodities can be reduced to a problem known as the "2-square routing" [11]:

- Input: A grid A of size $(\Theta(m) \times \Theta(m))$, with $m \le \frac{N}{8}$ being some integer; two disjoin sub-grids Q, Q' of A, of size $(m \times m)$ each, such that the minimum L_∞ distance between a vertex in Q and a vertex in Q' is $\Omega(m)$; commodities $M_i = (s_i, t_i)$ such that $s_i \in Q$ and $t_i \in Q'$.
- Output: A node-disjoint routing between each pair of M_i.

Subsequently, an $O(n^{1/4} \cdot \log n)$-approximation based on LP-rounding was proposed for the reduced problem.

In the case of "bad" commodities, a dynamic-programming based approach could be employed by grouping commodities in 16 groups according to the edges of G which their source and sinks are closest to. In each group, a dynamic programming procedure can be employed to route at least $OPT_{group}/ \le 4\sqrt{N} + 5$ commodities. The best result from the 16 groups is then used as the final result for all bad commodities, resulting in a total approximation ratio of $\frac{1}{16}(4\sqrt{N} + 5) = O(n^{1/4})$. The subproblem for "bad" commodities was explored further in [50].

The final approximation algorithm applies both approaches to both the good and the bad set of commodities and returns the better solution, leading to an $O(n^{1/4} \cdot \log n)$ overall approximation ratio. This is the best result for NDP in grids so far; unfortunately, many of the assertions in this method are not true for general multi-commodity flow problems, and the result would probably not lend itself well to generalisation. Also note that this method does not rely on rounding a fractional LP solution, and was thus able to exceed NDP's integrality gap of $O(\sqrt{|V|})$ [50]: indeed many recent breakthroughs on integral multi-commodity flow relied on breaking from that general approach.

5 Rounding techniques

We next proceed to introduce a number of common rounding techniques used in approximating integral multi-commodity flow or its associated problems. Earlier works tend to give more focus to this area, and numerous polynomial-time rounding schemes with a constant or $1+\epsilon$ factor (with respect to the integrality gap) were discovered early on.

5.1 Region growing

This is a technique that can be used to calculate a multi-cut solution from its LP relaxation, which as we discussed above, is equivalent to the LP formulation for multi-commodity flow. The basic intuition is to interpret the LP solution as a distance metric, and to divide the graph into regions with a certain radius.

This technique relies on the following definitions and lemma [2]:

Definition 5.1 *Let y_S be a variable associated with each subset $S \subseteq V$. Before the start of the region growing process, $y_S = 0$ for all S.*

Definition 5.2 *The radius of a subset A of G: $rad(A) = \sum_{S \subseteq A} y_S$*

Definition 5.3 *Let ΔA be the cut associated with a set $A \subseteq V$ (the set of edges with exactly one endpoint in A). Let $C_{\Delta(A)}$ be the total cost of the cut $\sum_{e \in \Delta(A)} c(e)$.*

Lemma 5.4 *Let A be a set where $C_{\Delta(A)} \leq \epsilon \cdot wt(A)$, then $Rad(A) < \frac{\ln(k+1)}{\epsilon}$. In fact, one such value for A can be found in polynomial time.*

Note this formulation is slightly different from the one used in the lecture notes, but contains the same results. The notable difference is that the lecture notes the return value from the region-growing lemma is a distance (in terms of the LP solution as distance metric), whereas $Rad(A)$ is a "thickness" (counts the number of vertices within a region). It should be fairly obvious to recognise a $O(|E|)$ reduction between them.

We then move on to the algorithm. Given a fractional LP solution $flow_e^i, e \in E, i \in [1..k]$, this technique involves the following:

1. Make a copy G_1 of G which will be updated throughout this algorithm. Go through the sources from s_1 to s_k.
2. For each source s_i, $B_{s_i} \leftarrow \{s_i\}$, $y_{\{s_i\}} = 0$. We then keep adding the nearest vertex v in G_1 (in terms of the LP solution as distance, with any pathfinding algorithm) to B_{s_i}, remove v from the graph G_1, and finally we assign $d(s_i, v)$ to $y_{\{s_i\}}$. We repeat until $C_{\Delta(B_{s_i})} \leq \epsilon \cdot wt(B_{s_i})$.
3. Return the set of disjoint sets and calculate min-cut based on such a decomposition.

[2] gave the approximation ratio for this technique as $O(logk)$, and the running time of this routine as $O(|E| + |V| \log |V|)$.

5.2 Rounding by reduction to another problem

This is a technique mentioned in [1] which rounds a fractional LP solution by reducing it to another problem with a simple rounding scheme. Let Γ be some positive value. The following was proposed in [18] for single-source flows and generalised in [1]:

Lemma 5.5 *Given a fractional solution y for a set of constraints $\max wx : A[d]x \leq b$ where w belongs to a closed collection of vectors W, we can find, in polynomial time, an integer solution y' such that $wy' \geq wy/11.542\Gamma$ and $A[d]y' \leq b$ provided we have an oracle as the following:*

Given a fractional solution z to $\max w'x : Ax \leq b'$ for any $w' \in W$ and integer vector b', the oracle outputs an integer solution z' such that $wz' \geq wz/\Gamma$ and $Az' \leq b'$.

What this essentially does is reducing a more general flow problem (the one with arbitrary demands) to a more specific one (unit demands), the latter of which has a known integrality gap (4) which can be used as the value for Γ. The number 11.542 is an artefact of the technique used for the above lemma: the method has an integrality gap upper-bounded by $\frac{1}{\min(\epsilon(1-2\sqrt{\beta}+\beta),(1-\epsilon)\beta/2)}$, with $0 \leq \epsilon \leq 1$ and free choices of α and β between 0 and 1; $\alpha = \beta = 1/3$ would put an upper bound of the integrality gap at 12, while the best possible value that could be achieved is 11.542.

5.3 Randomised rounding

This method is credited to Raghavan and Tompson in [23]. The idea itself is very straightforward on a high level: for an LP relaxiation where each of the solution variables x_i take values between 0 and 1, use x_i as a parameter to generate the corresponding integral solution variable z_i by setting $z_i \leftarrow 1$ with a probability of x_i and $z_i \leftarrow 0$ with a probability of $1 - x_i$. Repeat until the result is feasible. Of course, this method can easily be adapted to scenarios where the solution variables take on values beyong $0, 1$ by rounding x_i into the nearest integer randomly, using the fractional part of x_i as the probability parameter.

An obvious potential problem with this method is the relatively high probability of each rounding being infeasible. For each constraint of the form $\sum_i a_i \cdot x_i < b_i$, the probability of $\sum_i a_i \cdot z_i > (1+\delta)b_i$ is shown to be bound by $(\frac{e^{\delta}}{(1+\delta)^{1+\delta}})^{b_i}$ [24].

While the expected number of repetitions needed is polynomial to n, that number can be substantially reduced by multiplying x_i by a constant $0 < \delta < 1$. By Markov's inequality, such a change would reduce the probability of each constraint being infeasible by a factor of at least $1/\delta$ [24].

5.3.1 Randomised rounding on trees

This is a rounding technique which only works when commodities can be sorted in such a way that any loopless source-sink path in a later commodity does not intersect with a previous commodity [4]. Examples of such graphs include lines, rings and trees.

For some constant a and an LP solution $flow_e^i, e \in E, i \in I$ (using the compact LP formulation), we do the following:

1. Root the tree arbitrarily. For each commodity $C_i = (s_i, t_i), i \in [1..k]$, define the depth $l(C_i)$ as the depth of the least common ancestor of s_i and t_i.

2. Independently select a subset of B of $[1..k]$, where each $i \in [1..k]$ is selected with a probability $\frac{flow^i_{(t_i,s_i)}}{a}$. Sort B in non-decreasing order of depth (as defined above).

3. Let the set of solutions $S = \emptyset$. Go through B in sorted order. For each commodity $C_i \in B$, add C_i to S if doing so does not violate the feasibility of the current set of commodities in S, discard C_i otherwise.

4. Return S.

This technique was first proposed by Calinescu, Chakrabarti, Karloff and Rabani [4] for lines with unit capacity. It was analysed in [8] where it was found that, as long as the edge capacities are integers, this technique has a constant factor approximation ratio (the original work limited the graph type to just lines and rings, although [1] pointed out that the same argument works for trees).

The value for a chosen in [1] was 31.25, and a further lemma was proved stating that the chance for any commodity to be rejected after it has been picked given $a = 31.25$ is .597. The approximation ratio of this rounding method was thus given to be $\frac{1}{(1-.597)a} \leq 77.51$. Such a number could be considered quite large when considering the existence of $O(logn)$ and $O(longk)$ rounding schemes (and the fact that n and k could be small in some cases).

While this technique may feel similar to the general randomised random introduced above, note that this method is guaranteed to return a feasible, constant-factor result in the first iteration, as opposed to requiring a Chernoff-style bound on the number of iterations; this is achieved at the expense of only being applicable to trees and few other limited types of graphs.

5.4 Oblivious (greedy) rounding

This is a method which allows for an integral solution to be obtained without first having to solve the LP relaxation, first proposed by Young in [22]. All of the rounding schemes discussed above involved using the optimal solution to a fractional LP, which in cases such as randomised rounding, could prove to dominate the time complexity [23].

This method relies on the following oracle: given a convex set P in $\mathbb{R}^n$ (ie. the solution set to an LP with n variables), f a linear function from P to $\mathbb{R}^m$ (ie. the LP constraints), and non-negative $y \in \mathbb{R}^m$, the oracle returns x and $f(x)$ where x minimises $\sum_j y_j f_j(x)$ (a shortest path). This oracle is then employed iteratively to send the next integral packet of flow of the least-prioritised commodity through the least congested path, after which the lengths of the edges on that path are increased. At the end of the algorithm the flow result is scaled down to keep it feasible. The number of iterations needed to get a $(1+\epsilon)$-approximation has quadratic dependence on ϵ^{-1} and on the width of the solution set (as a convex set).

Note the similarity of this method to the greedy algorithms. In fact, when applying the packing version of this method on set cover, we get the greedy set cover algorithm [22].

6 References

1. Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepard. Multicommodity demand flow in a tree and packing integer programs. ACM Trans. Algorithms 3, 2007.
2. Naveen Garg, Vijay Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. SIAM Journal on Computing, 25:698-707, 1996.
3. Peng Zhang, Daming Zhu, and Junfeng Luan. An approximation algorithm for the generalized k-Multicut problem. Discrete Applied Mathematics 160:1240-1247, 2012.
4. Gruia Calinescu, Amit Chakrabarti, Howard Karloff, Yuval Rabani. Improved approximation algorithms for resource allocation. Proc. of 9th IPCO, 401-414, Springer-Verlag LNCS, 2002.
5. Chandra Chekuri, Sanjeev Khanna, F. Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. ACM Symposium on Theory of Computing, P183-192, 2005.
6. Ken-ichi Kawarabayashi, Yusuke Kobayashi. All-or-nothing multicommodity flow problem with bounded fractionality in planar graphs. SIAM Journal on Computing, 47:1483-1504, 2013.
7. Neil Robertson, Paul D. Seymour, Robin Thomas. Quickly excluding a planar graph. J Combin. Theory Ser. B, 62:323-348, 1994.
8. Amit Chakrabarti, Chandra Chekuri, Anupam Gupta, Amit Kumar. Approximation algorithms for the unsplittable flow problem. Algorithmica, 47:53-78, 2007.
9. Julia Chuzhoy. Flows, cuts and integral routing in graphs - an approximation algorithmist's perspective. Proc. of the International Congress of Mathematicians, 585-607, 2014.
10. Michael Saks, Alex Samorodnitsky, Leonid Zosin. A lower bound on the integrality gap for minimum multicut in directed networks. Combinatorica, 24(3):525-530, 2004.
11. Julia Chuzhoy, David H. K. Kim. On approximating node-disjoing paths in grids. Proc. of APPROX 2015, LIPIcs-Leibniz International Proceedings in Informatics (Vol. 40), 2015.
12. Chandra Chekuri, Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. Journal of the ACM, 63(5), Article 40, 2016.
13. Tom Leighton, Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation alorithms. Journal of the ACM, 46:787-832, 1999.
14. Sanjeev Arora, Satish Rao, Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. Journal of the ACM, 56(2), 2009.
15. Alina Ene, Matthias Mnich, Marcin Pilipczuk, Andrej Risteski. On routing disjoint paths in bounded treewidth graphs. SWAT: 15th Scandinavian Symposium and Workshops on Algorithm Theory, 53:1-15, 2016.
16. Daniel Dressler, Martin Strehler. Polynomial-time algorithms for special cases of the maximum confluent flow problem. Discrete Applied Mathematics, 163:142-154, 2016.

17. Lester Randolph Ford, Delbert Ray Fulkerson. Flows in networks. Princeton University Press, Princeton, NJ, 1962.
18. Stavros Kolliopoulos, Clifford Stein. Approximating disjoin-path problems using greedy algorithms and packing integer problems. Mathematical Programming A, (99), 63-87, 2004.
19. Joseph Cheriyan, Howard Karloff, Yuval Rabani. Approximating directed multicuts. Combinatorica, 25(3):251-269, 2005.
20. Julia Chuzhoy, Sanjeev Khanna. Polynomial flow-cut gaps and hardness of directed cut problems. Journal of the ACM, 56(2):6, 2009.
21. Stefan Arnborg, Derek G. Corneil, Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. SIAM Journal on Algebraic Discrete Methods, 8(2):277-284, 1987.
22. Neil E. Young, Randomized rounding without solving the linear program. Sixth ACM-SIAM Symposium on Discrete Algorithms (SODA95), 1995.
23. Prabhakar Raghavan, Clark D. Tompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. Combinatorica 7, 365-374, 1987.
24. Prabhakar Raghavan. Porbabilistic construction of deterministic algorithms: Approximating packing integer programs. Jounal of Computer and System Sciences, 37(2):130-143, 1988.
25. Loïc Séguin-Charbonneau, F. Bruce Shepherd. Maximum edge-disjoint paths in planar graphs with congestion 2. Proc. of 52nd IEEE Symposium on Foundations of Computer Science (FOCS), 200-209, 2011.
26. Haruko Okamura, Paul D. Seymour. Multicommodity flows in planar graphs. Journal of Combinatorial Theory, B(31): 75-81, 1981.
27. Chandra Chekuri, Sanjeev Khanna, F. Bruce Shepherd. Edge-disjoin paths in planar graphs with constant congestion. SIAM Journal on Computing, 39(1):281-301, 2009.
28. Julia Chuzhoy, Shi Li. A Polylogarithimic Approximation Algorithm for Edge-Disjoint Paths with Congestion 2. Journal of the ACM (JACM), 63(5):1-51, 2016.
29. Naveen Garg, Vijay Vazirani, Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica, 18:3-20, 1997.
30. Michael X. Goemans, David P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. Approximation algorithms for NP-hard problems, 144-191, 1996.
31. Naveen Garg, Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. IEEE Symposium on Foundations of Computer Science, 300:309, 1998.
32. Lisa Fleischer. Approximating fractional multicommodity flows independent of the number of commodities. SIAM Journal of Discrete Math., 13(4):505–520, 2000.
33. Mohamed Bouklit, David Coudert, Jean-François Lalande, Christophe Paul, Hervé Rivano. Approximate multicommodity flow for WDM networks design. In Sirocco 10, Umea, Sweden, June 2003.
34. David Coudert, Hervé Rivano, Xavier Roche. A combinatorial approximation algorithm for the multicommodity flow problem. International Workshop on Approximation and Online Algorithms (WAOA'03), 193-230, 2003.
35. George Karakostas. Faster approximation schemes for fractional multicommodity flow problems. ACM Transactions on Algorithms 4, Article No.13, 2008.
36. Farhad Shahrokhi, David W. Matula. The maximum concurrent flow problem. Journal of the ACM, Vol. 37, Article No.2, 1990.
37. Aleksander Mądry. Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. Proc. of the 42nd ACM Symposium on Theory of Computing, 121-130, 2020.
38. Lester Randolph Ford, Delbert Ray Fulkerson. Constructing maximal dynamic flows from static flows. Oper. Res. 6(3):419-433, 1958.
39. Salman Khodayifar. Minimum costmulticommodity network flow problem in time-varying networks: by decomposition principle. Optim Lett 15, 1009–1026, 2021.
40. Enrico Grande, Gaia Nicosia, Andrea Pacifici, Vincenzo Roselli. An exact algorithm for a multicommodity min-cost flow over time problem. Electron. Notes Discret. Math. 64: 125-134, 2018.

41. Bettina Klinz, Gerhard J. Woeginger. Minimum-cost dynamic flows: The series-parallel case, Networks 43, 153–162, 2004.
42. Lisa Fleischer, Martin Skutella. Minimum cost flows over time without intermediate storage. Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 03, 66-75, 2003.
43. Dritan Nace, Michał Pióro. Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial. IEEE Communications Surveys and Tutorials, 10(4):5-17, 2008.
44. Philip Klein, Neil E. Young. On the number of iterations for dantzig-wolfe optimization and packingcovering approximation algorithms. Proc. of the 7th Internation IPCO Conference, 320-327, 2002.
45. Yurii Nesterov. Smooth minimization of non-smooth functions. Mathematical Programming, 103(1):127–152, 2005.
46. Jacek Gondzio, Robert Sarkissian. Column generation with a primal-dual method. Technical Report 96.6, Logilab, 1996.
47. Jacek Gondzio, Pablo González-Brevis, Pedro Munari, New developments in the primal–dual column generation technique, European Journal of Operational Research, 224(1): 41-51, 2013.
48. D. Bienstock and G. Iyengar. Solving fractional packing problems in $\tilde{O}(1/\epsilon)$ iterations. Proc. of the 36th Annual ACM Symposium on Theory of Computing, 146–155, 2004.
49. Yurii Nesterov. Fast gradient methods for network flow problems. 20th International Symposium of Mathematical Computing, 2009.
50. Julia Chuzhoy, David Kim, Rachit Nimavat. Improved approximation algorithm for node-disjoint paths in grid graphs with sources on grid boundary. arXiv:1805.09956 (Preprint), 2018.
51. Jochen Könemann, Ojas Parekh, David Pritchard. Multicommodity flow in trees: packing via covering and iterated relaxation. Algorithmica, 68:776-804, 2014.
52. Tom Leighton, Fillia Makedon, Serge Plotkin, Clifford Stein, Éva Tardos, Spyros Tragoudas. Fast approximation algorithms for multicommodity flow problems. Journal of Computer and System Sciences, 50(2):228-243, 1995.
53. Pierre-Olivier Bauguion, Claudia D'Ambrosio, Leo Liberti. Maximum concurrent flow with incomplete data. Combinatorial Optimization. ISCO 2018. Lecture Notes in Computer Science, vol 10856, 2018.
54. Anupam Gupta, Kunal Talwar, David Witmer. Sparsest cut on bounded treewidth graphs: algorithms and hardness results. Proc. of the 45th annual ACM Symposium on theory of computing, 281-290, 2013.
55. Sanjeev Arora, Rong Ge, Ali Kemal Sinop. Towards a better approximation for sparsest cut? IEEE 54th Annual Symposium on Foundations of Computer Science, 270-279, 2013.
56. Venkatesan Guruswami, Sanjeev Khanna, Rajmohan Rajaraman, Bruce Shepherd, Mihalis Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. Journal of Computer and System Sciences, 67(3):473-496, 2003.
57. Chandra Chekuri, Sanjeev Khanna, Bruce Shepherd. The all-or-nothing multicommodity flow problem. SIAM Journal on Computing, 42(4):1467-1493, 2013.
58. Weibin Dai, Jun Zhang, Xiaoqian Sun. Chinese Journal of Aeronautics, 30(4):1481-1492, 2017.
59. Haruko Okamura, Paul D. Seymour. Multicommodity flow in planar graphs. J Combin. Theory Ser. B, 31(1):75-81, 1981.
60. James R. Lee, Manor Mendel, Mohammed Moharrami. A node-capacitated Okamura-Seymour theorem. Mathematical Programming, 153:381-415, 2015.
61. Krzysztof Fleszar, Matthias Mnich, Joachim Spoerhase. New algorithms for maximum disjoint paths based on tree-likeness. Mathematical Programming, 171:433-461, 2018.