Parameterized Maximum Path Coloring

Advanced Topics on
Algorithms & Complexity

Avarikioti Zeta

June 19, 2014

Path Coloring Problems

Path Coloring (PC)

Input: *Graph G, set of paths R*

Output: *Coloring of the paths such that any two paths that share the same edge have distinct colors and the number of colors used is minimized*

Maximum Path Coloring (MaxPC)

Input: *Graph G, number of colors W*

Output: *Maximum cardinality set of paths properly colored*

Maximum Path Coloring (MaxPC) in trees

Input: *undirected tree G=(V,E), multi-set of demands D⊆V*V (paths in G), number of colors W, number of demands we seek to satisfy B*

Output: *W mutually disjoint subsets $D_1, D_2 \ldots D_w \subseteq D$ s.t. no set D_i contains two demands that share an edge and* $\sum_{i=1}^{W} |Di| \geq B$

Other problems we will need...

Maximum Edge Disjoint Paths (MaxEDP)

Input: *Graph G, w=1 (one color)*

Output: *Maximum cardinality set of paths properly colored*

Capacity Maximum Path Coloring (CapMaxPC)

Input: *Graph G, number of colors W, edge capacities* $1 \leq c(e) \leq W$

Output: *Maximum cardinality set of paths properly colored*

Disjoint Neighborhoods Packing (DNP)

Input: undirected *graph* $G=(V,E)$

Output: *Maximum cardinality set of* $V' \subseteq V$ *such that* $\forall u,v \in V'$ *we have* $N(u) \cap N(v) = \emptyset$

Previous work for Path Coloring

PC is NP-hard for:

- undirected stars (edge coloring in multi-graphs)
- Undirected rings (coloring circular-arc graphs)
- Bi-directed binary trees

PC is FPT for:

- Undirected trees when parameterized by the maximum degree of the tree Δ
- Bi-directed trees when parameterized by the maximum number of request touching any node

- 4/3- approximation algorithm for undirected trees
- 5/3-approximation algorithm for bi-directed trees

Previous work for Maximum Path Coloring

XP algorithm, not FPT

MaxPC is solvable in polynomial time, in bi-directed trees, if both maximum degree and number of colors are constant (straightforward extension in undirected trees)

MaxEDP (MaxPC for W=1)

- Is NP-hard for bi-directed trees
- Is P for undirected trees

- 1,58- approximation algorithm for undirected trees
- 2,22-approximation algorithm for bi-directed trees

Contribution of this paper

- Parameterization that does not involve the objective function-W, Δ, t:
  - ✓ None of the parameters can be removed from the exponent of n, under standard complexity assumptions, even if the others are small constants
  - ✓ MaxPC is NP-complete even in binary trees
  - ✓ Gap between PC and MaxPC

many rejected requests in reductions, in contrast with PC

- Parameterization that involves the objective function-W, Δ,T (number of requests rejected):
  - ✓ FPT in undirected and bi-directed trees with parameters W, Δ, T
  - ✓ FPT in binary trees when parameterized by T
  - ✓ FPT for undirected trees, rings and bi-directed trees when parameterized by the size of the solution (MaxEDP)

Structural Parameterization

Theorem 1: *(cW, $c\Delta$, ct)-MaxPC can be solved in polynomial time for both undirected and bi-directed trees*

Proof sketch

(Bottom-up dynamic programming on the tree decomposition of G)

- Root the tree decomposition on some arbitrary bag (the vertices of a non-leaf bag of the decomposition form a separator of G)
- In any feasible solution, for any given bag B, we can have only $O(W\Delta t)$ satisfied demands touching the vertices of B:
  - ✓ Every edge has at most W satisfied demands going through it
  - ✓ Every vertex has at most Δ edges touching it (2Δ for bi-directed trees)
  - ✓ All bags have at most t+1 vertices
- With standard treewidth technique we calculate bottom-up for each possible local solution, what is the maximum number of satisfied demands in the graph induced by the vertices in the bag and those below in tree decomposition

Structural Parameterization

Is there an FPT algorithm for (pW, pΔ, pt)-MaxPC ?

We will show that IS $\leq_{FPT}$ DNP $\leq_{FPT}$ CapMaxPC $\leq_{FPT}$ MaxPC ... *so* ***NO!***

Lemma 1: *For both undirected and bi-directed graphs we have*

a) (pW, cΔ)-CapMaxPC $\leq_{FPT}$ (pW, cΔ)-MaxPC

b) (cW, pΔ)-CapMaxPC $\leq_{FPT}$ (cW, pΔ)-MaxPC

c) (cW, cΔ, pt)-CapMaxPC $\leq_{FPT}$ (cW, cΔ, pt)-MaxPC

Proof

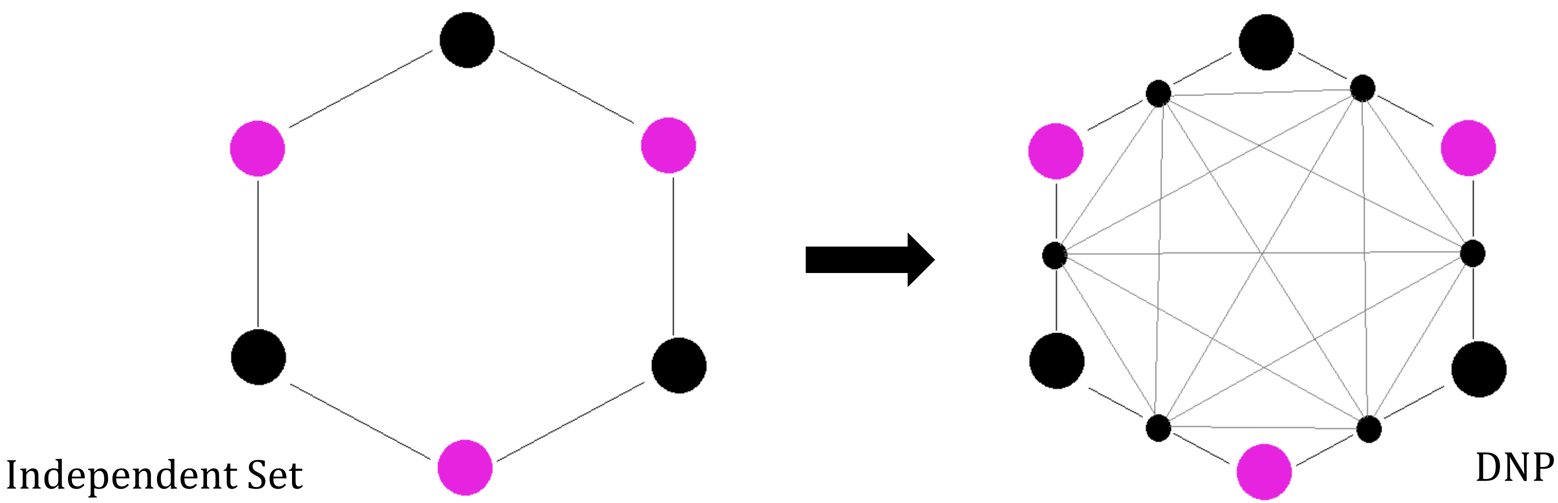
- For each edge (u, v) with capacity c<W we add W-c demands from u to v. Let A be the set of these new demands
- We set the capacity of edges W (instance of MaxPC)
- CapMaxPC instance has a coloring satisfying B demands iff the new MaxPC has a coloring satisfying A+B demands

Structural Parameterization

Lemma 2: DNP is W[1]-hard (IS $\leq_{FPT}$ DNP)

Proof sketch

Subdivide every edge of G=(V,E). Connect all newly added vertices in a clique. The new graph has a packing of k disjoint neighborhoods iff the original graph has an independent set of size k.



Structural Parameterization

Theorem 2: *(pW, cΔ)-MaxPC is W[1]-hard for both undirected and bi-directed trees*

Proof

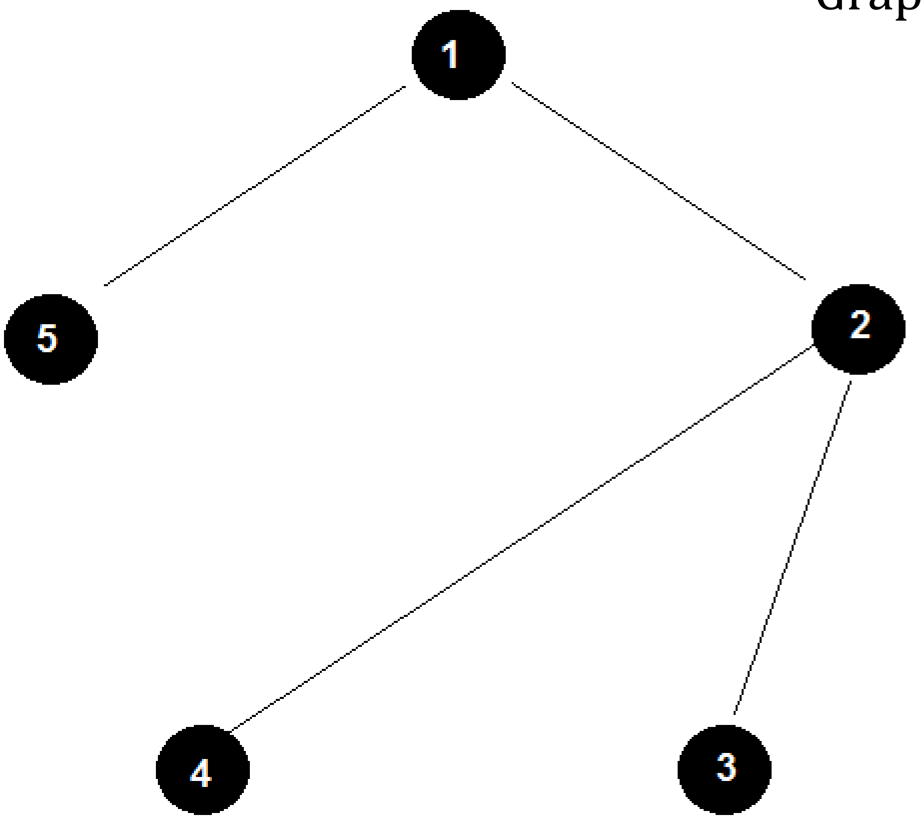
At first, we will prove the theorem for undirected graphs. We have an instance of DNP: a graph G=(V, E) and a target size for the DNP set k. We will start with the gadget construction (let |V|=n):

- Backbone: path of n+2 vertices, b_i, $0 \le i \le n+1$
- Branches: n+2 copies of a path with n vertices, p_{ij}, $0 \le i \le n+1$, $1 \le j \le n$
- Connect b_i to p_{i1}
- For each $1 \le i, j \le n$ we add three vertices in the graph w_{ij}, u_{ij}, v_{ij} and the edges (u_{ij}, w_{ij}), (v_{ij}, w_{ij}) and (w_{ij}, p_{ij})

Structural Parameterization

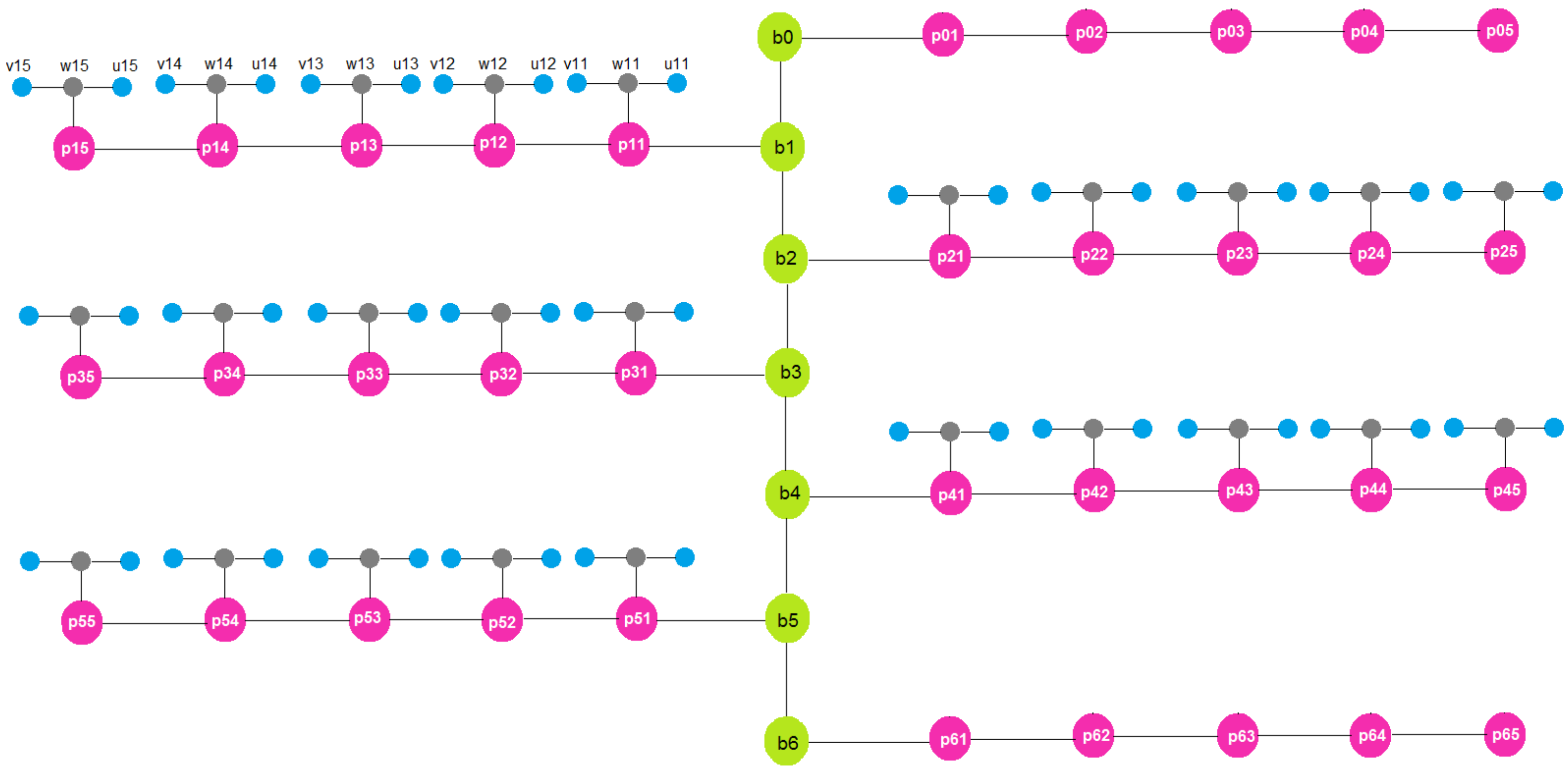
Graph G=(V, E)

N(1)={2,5}
N(2)={1,3,4}
N(3)={2}
N(4)={2}
N(5)={1}



n=5
k=2

Proof of Theorem 2



Now the graph is a tree of maximum degree 3

Proof of Theorem 2

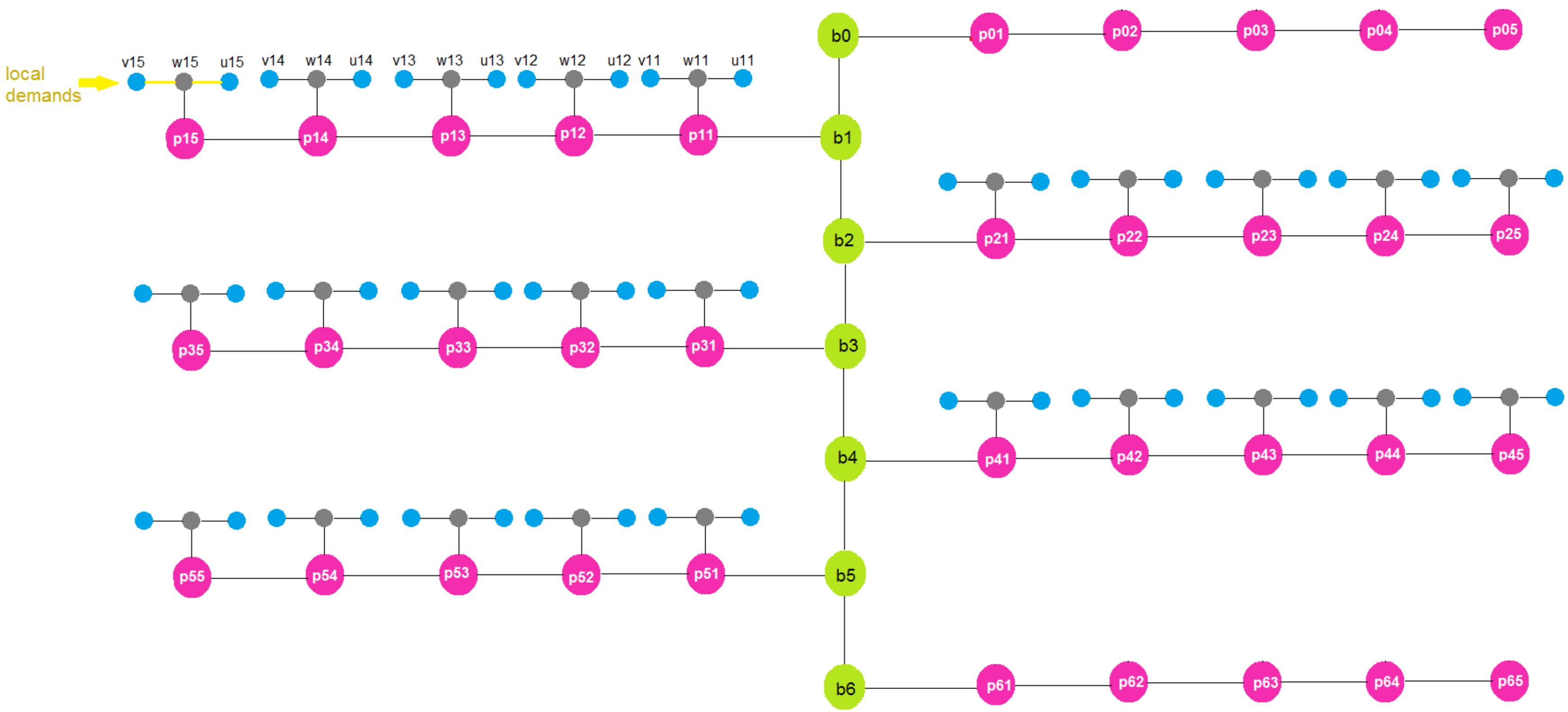
Demands & Capacities

Suppose that the vertices of the original graph are numbered $\{1,2,\ldots,n\}$. For each $i \in V$ we consider the closed neighborhood $N(i)$ in increasing order and let $N(i) = \{j_0, j_1, \ldots, j_{d(i)}\}$, where $d(i)$ is the degree of i.

- Global demands:
  - ✓ From $p_{0,i}$ to $u_{j_0,i}$
  - ✓ $\forall l,\ , 0 \le l \le d(i)$, from $v_{j_l,I}$ to $u_{j_{l+1},i}$
  - ✓ From $v_{j_{d(i)},i}$ to $p_{n+1,I}$
- Local demands: from from $v_{i,j}$ to $u_{i,j}$
- Capacities:
  - ✓ 2 for the edges $(b_i, p_{i,1})$
  - ✓ 1 for the edges (u_{ij}, w_{ij}) and (v_{ij}, w_{ij})

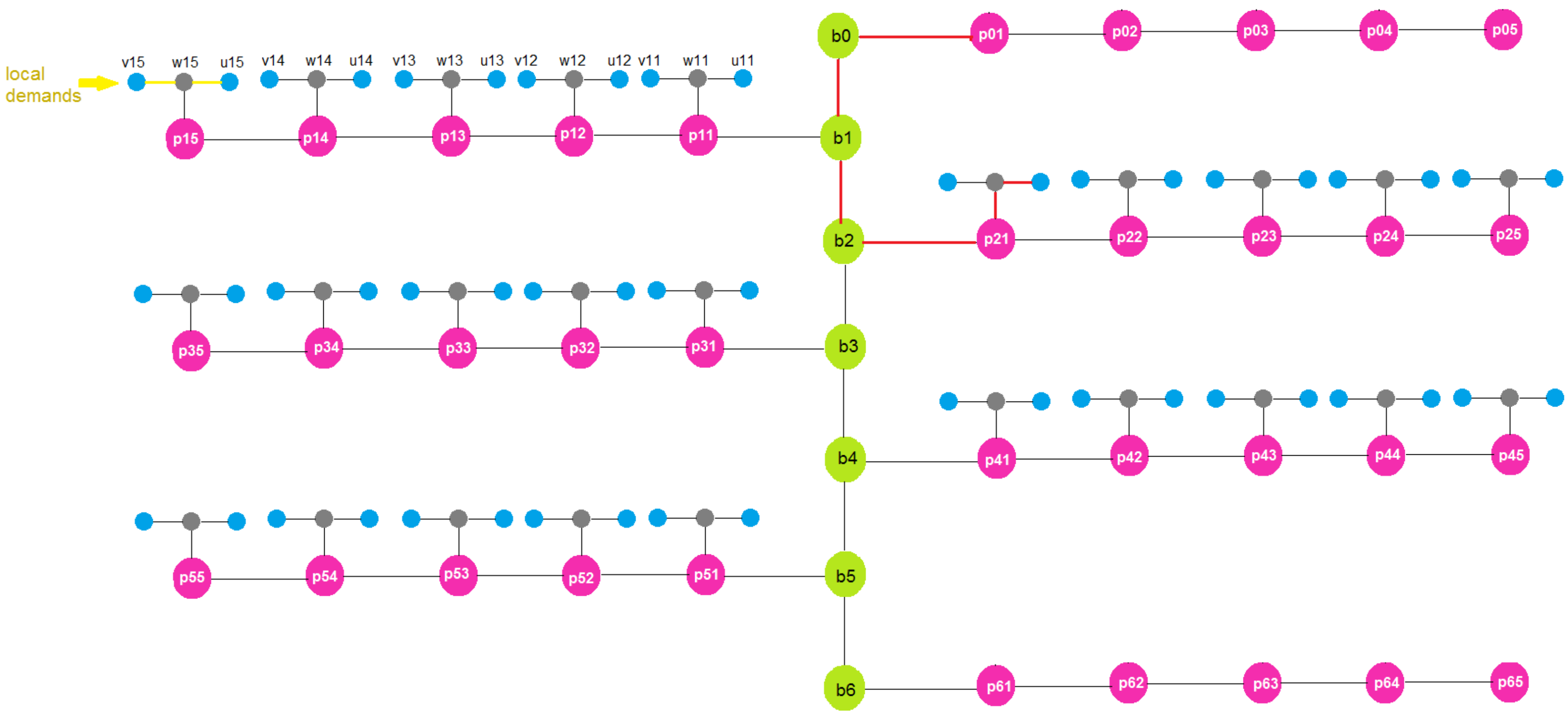
Proof of Theorem 2-demands

Local demands



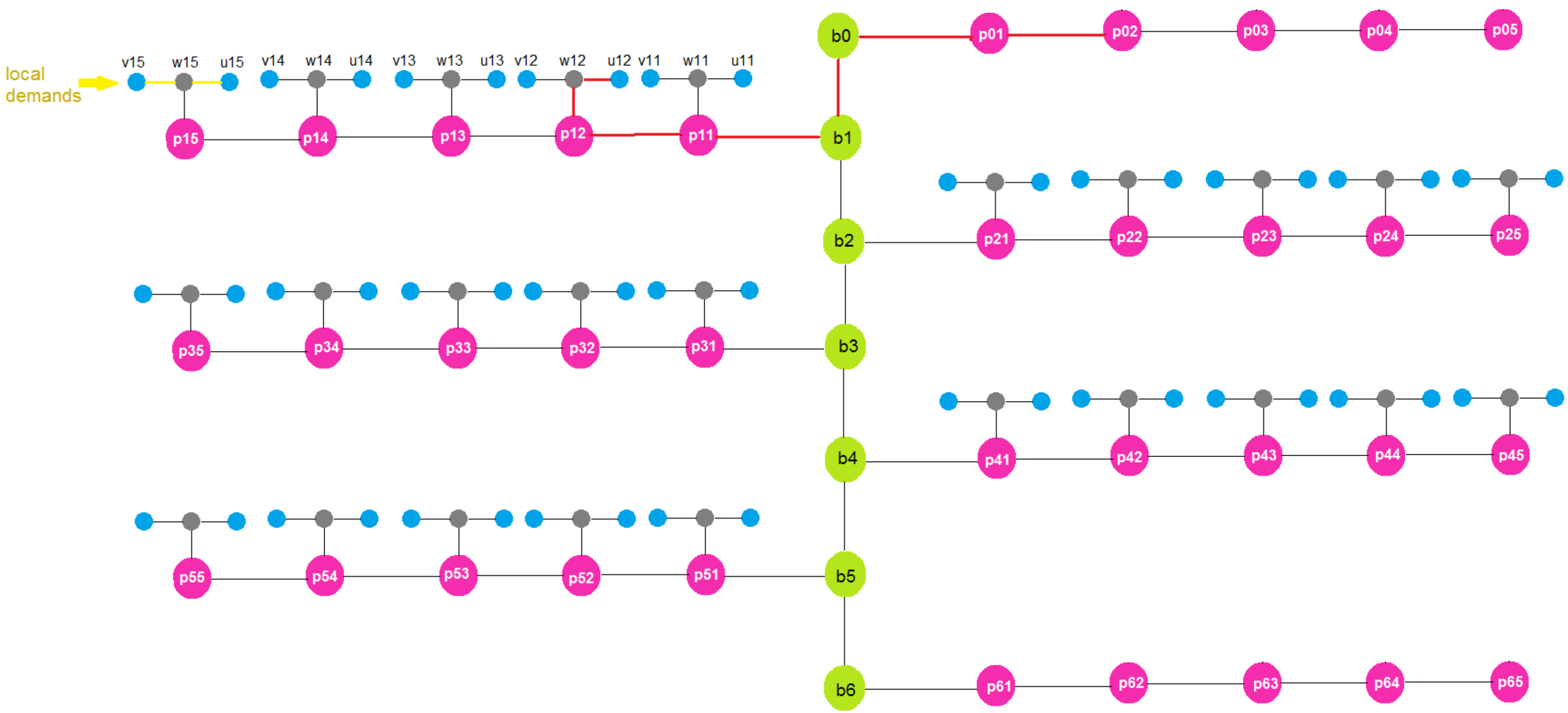
Proof of Theorem 2-demands

From $p_{0,i}$ to $u_{j_0,i}$



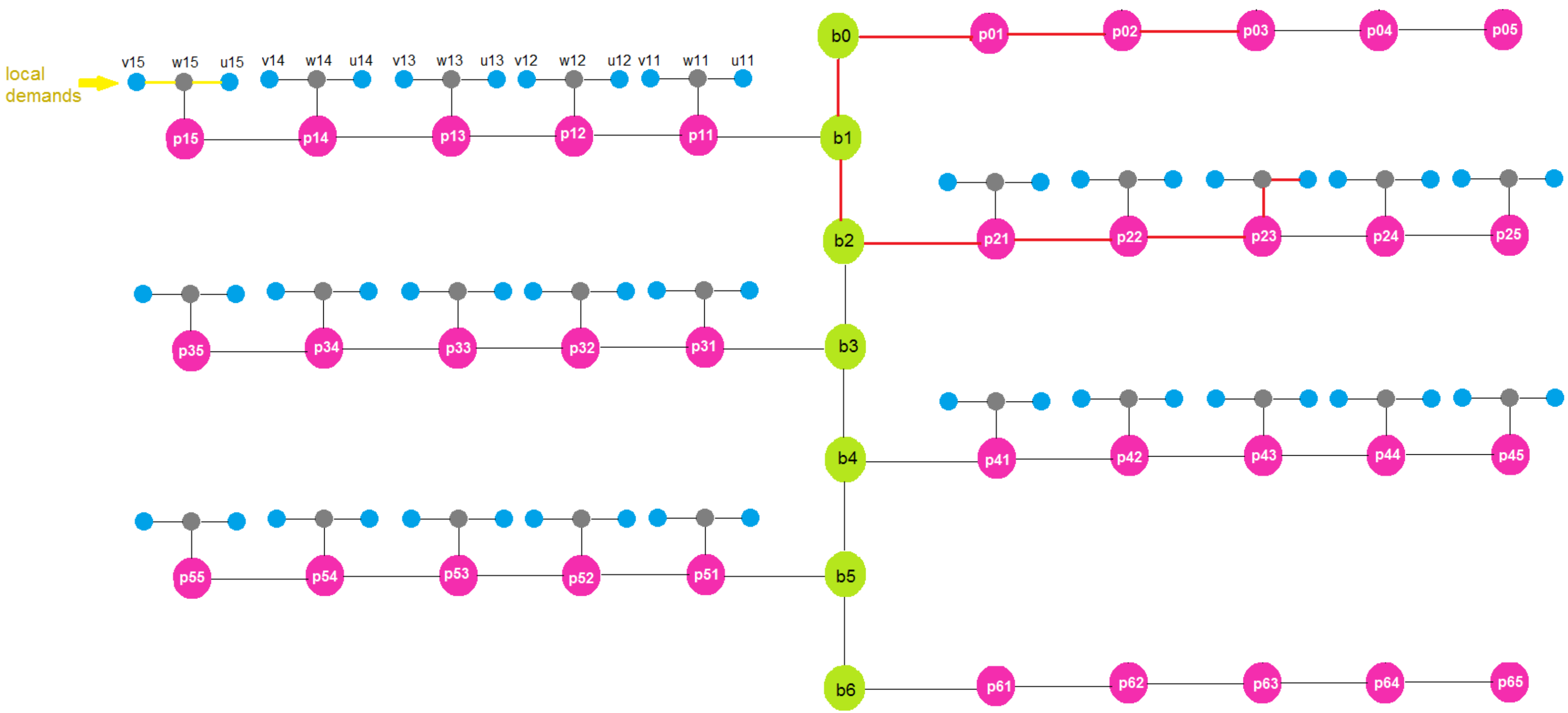
Proof of Theorem 2-demands

From $p_{0,i}$ to $u_{j_{0},i}$



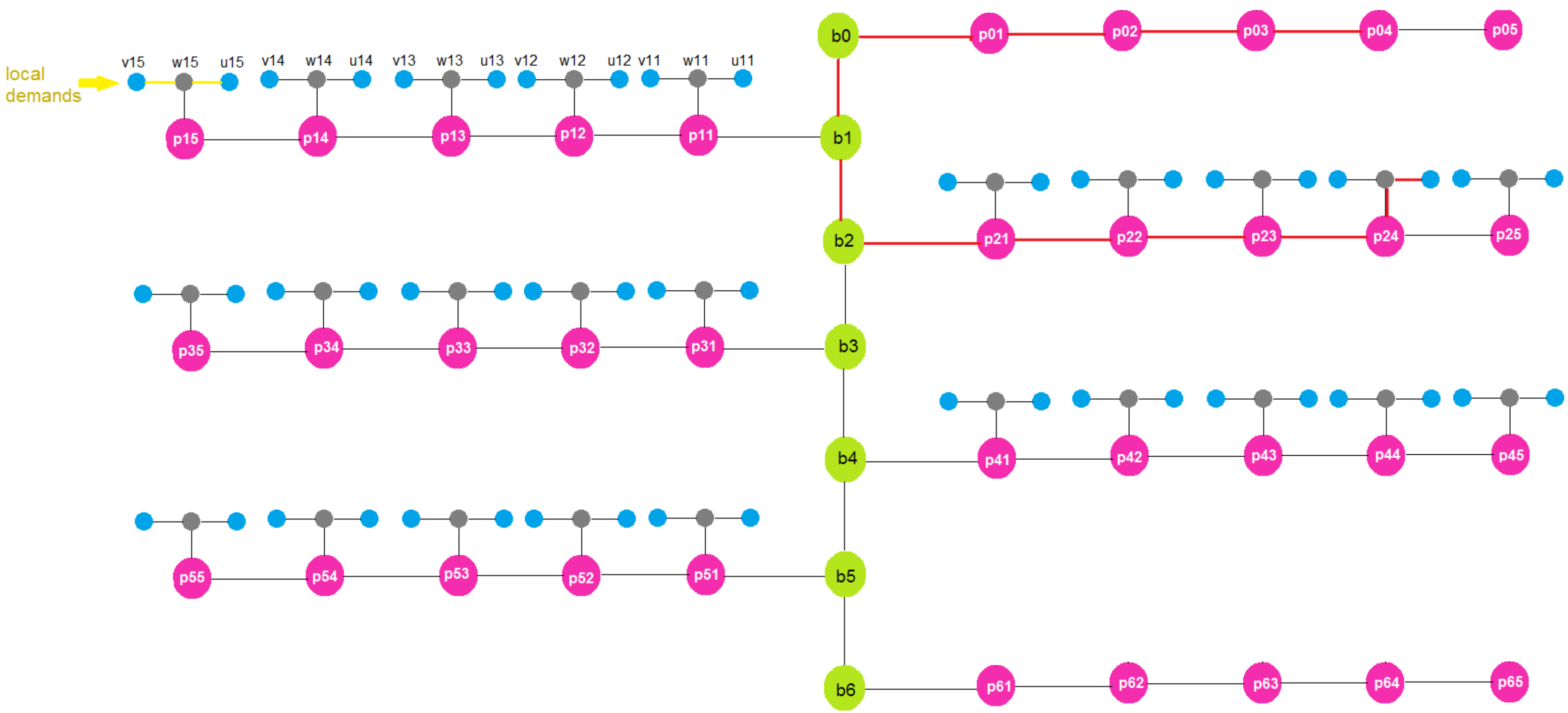
Proof of Theorem 2-demands

From $p_{0,i}$ to $u_{j_{0},i}$



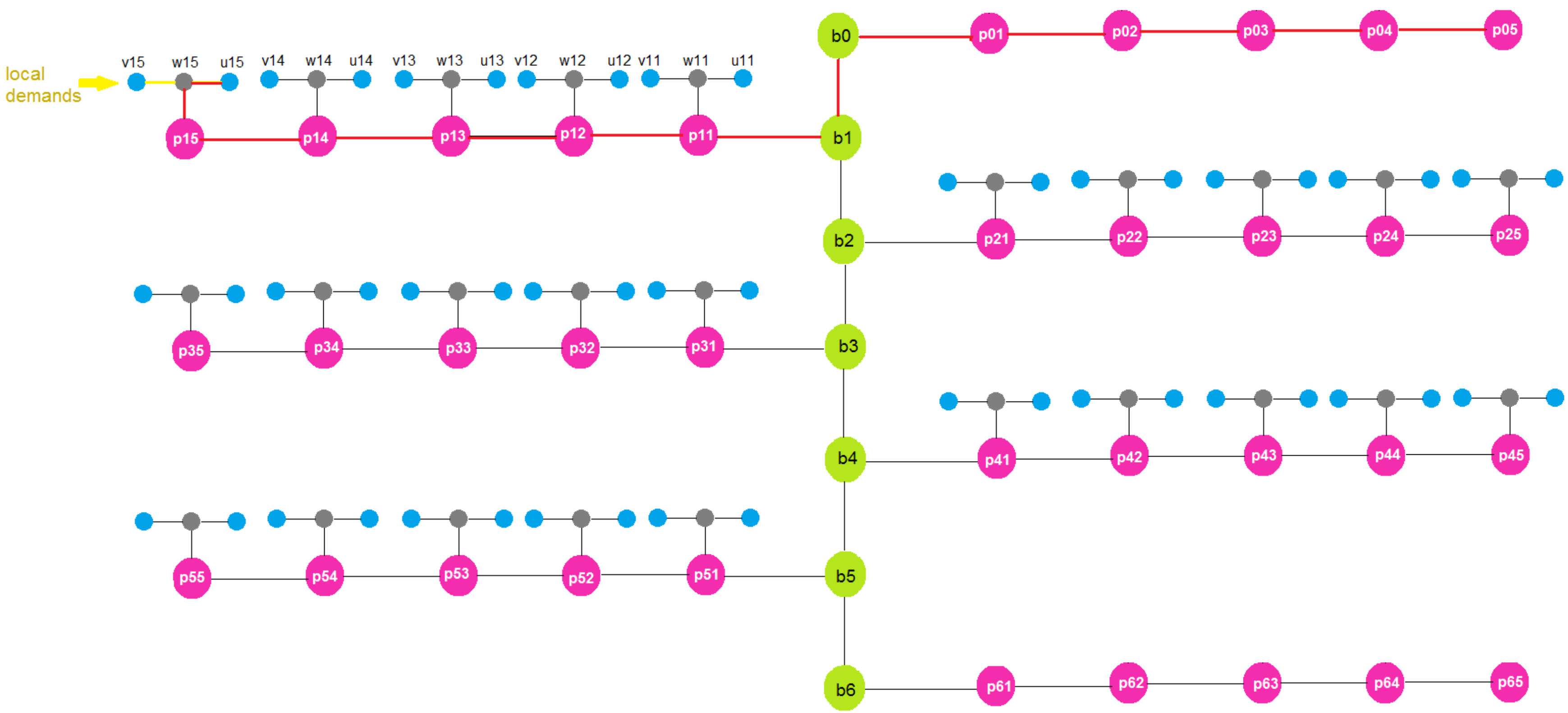
Proof of Theorem 2-demands

From $p_{0,i}$ to $u_{j_{0},i}$



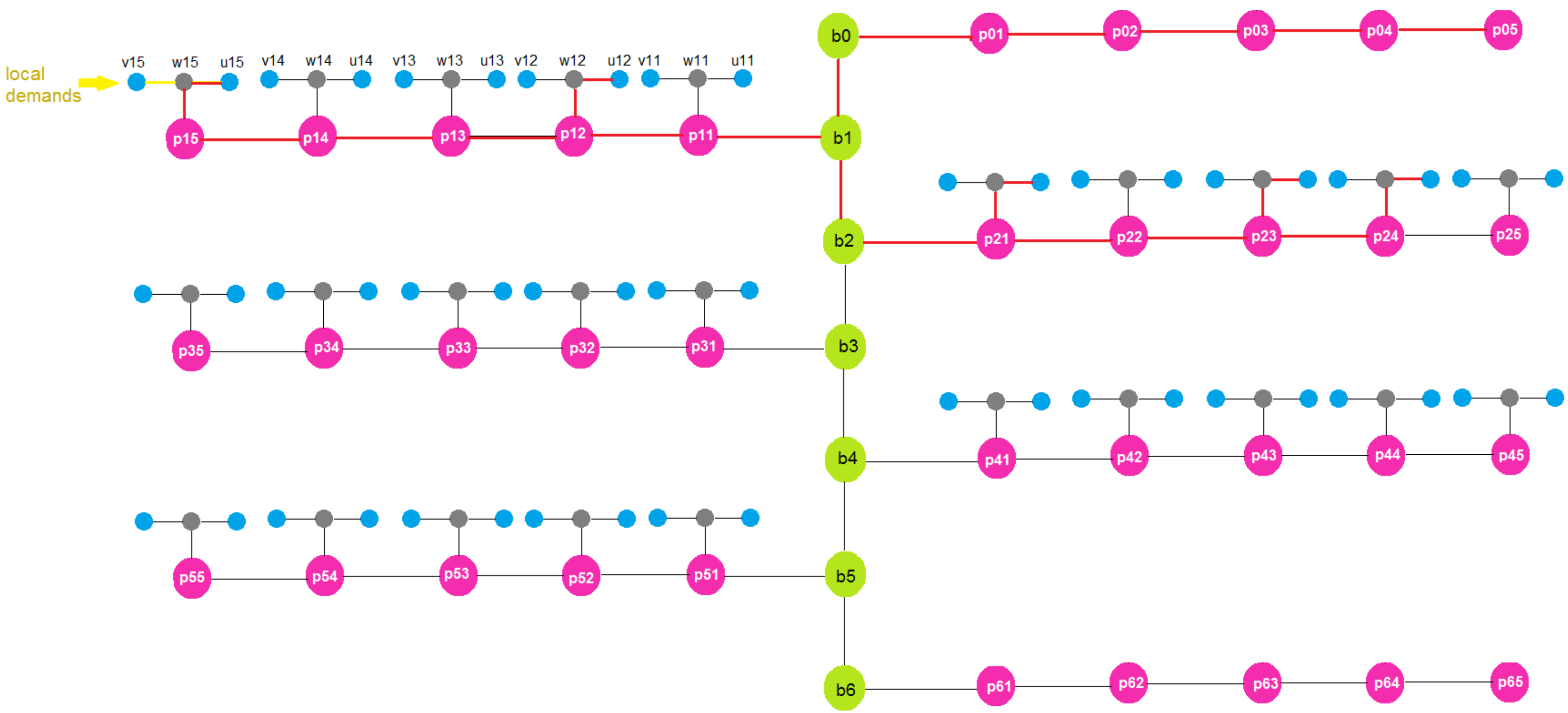
Proof of Theorem 2-demands

From $p_{0,i}$ to $u_{j_0,i}$



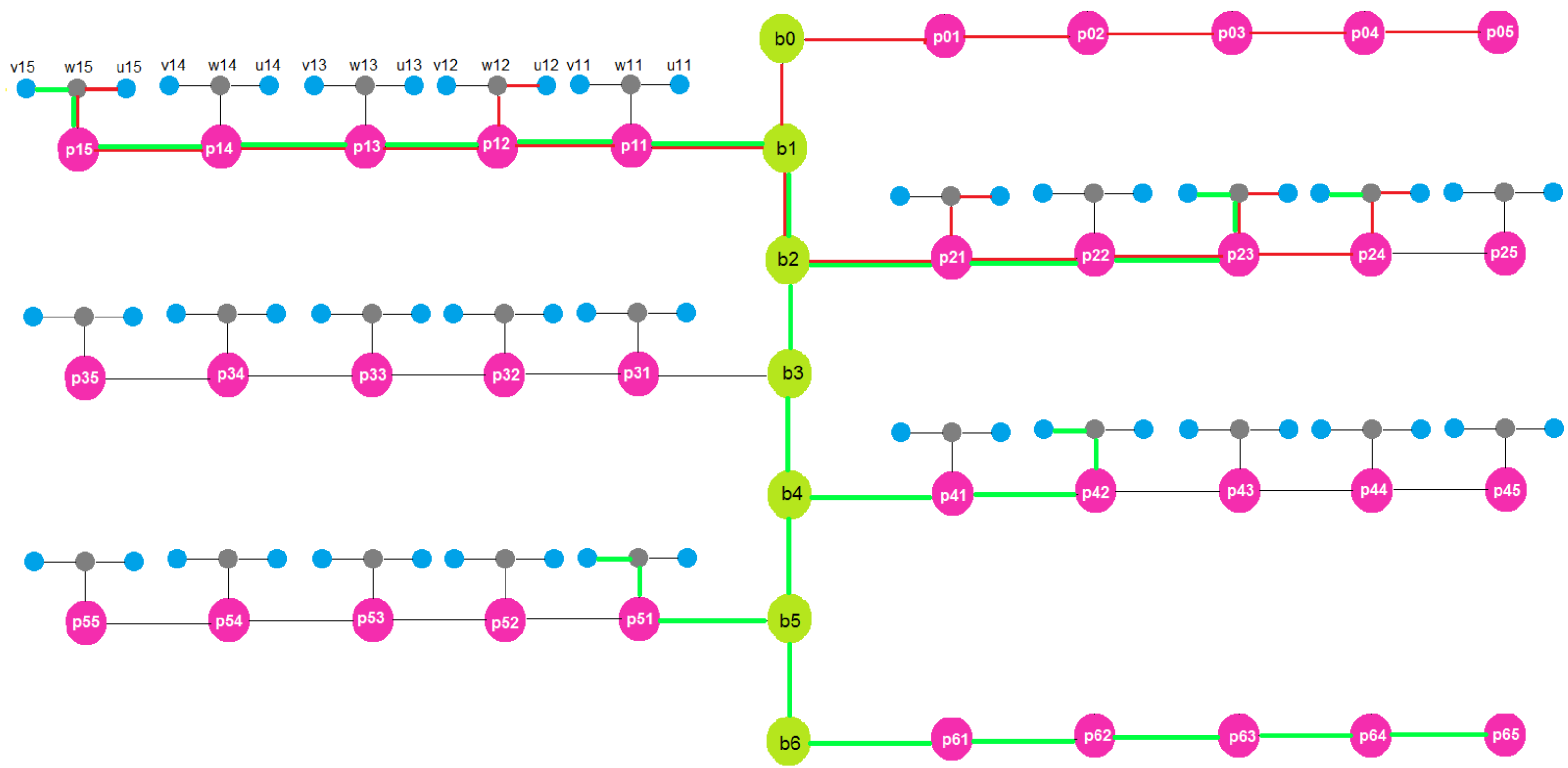
Proof of Theorem 2-demands

From $p_{0,i}$ to $u_{j_0,i}$



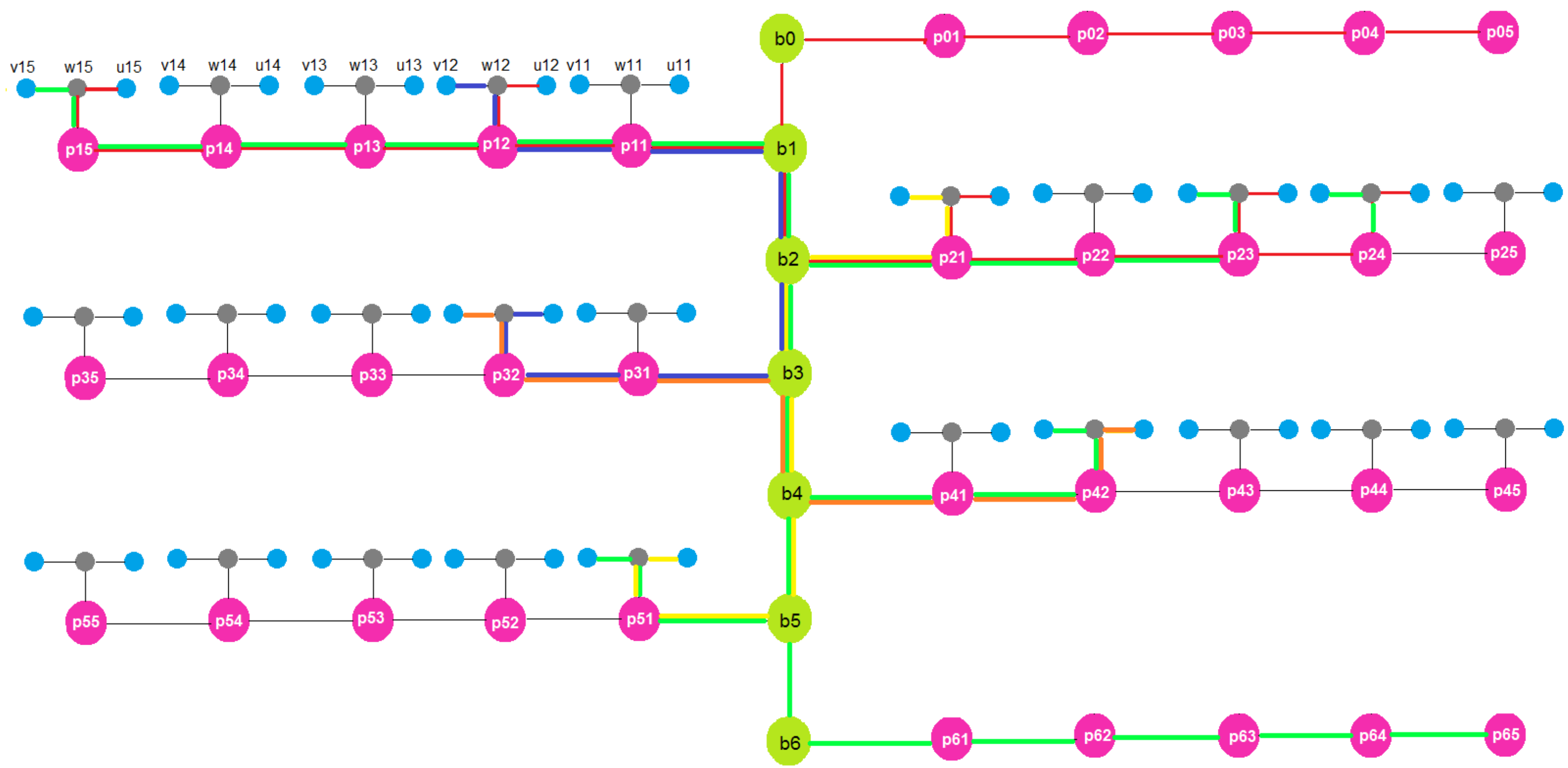
Proof of Theorem 2-demands

From $v_{j_{d(i)},i}$ to $p_{n+1,i}$



Proof of Theorem 2-demands

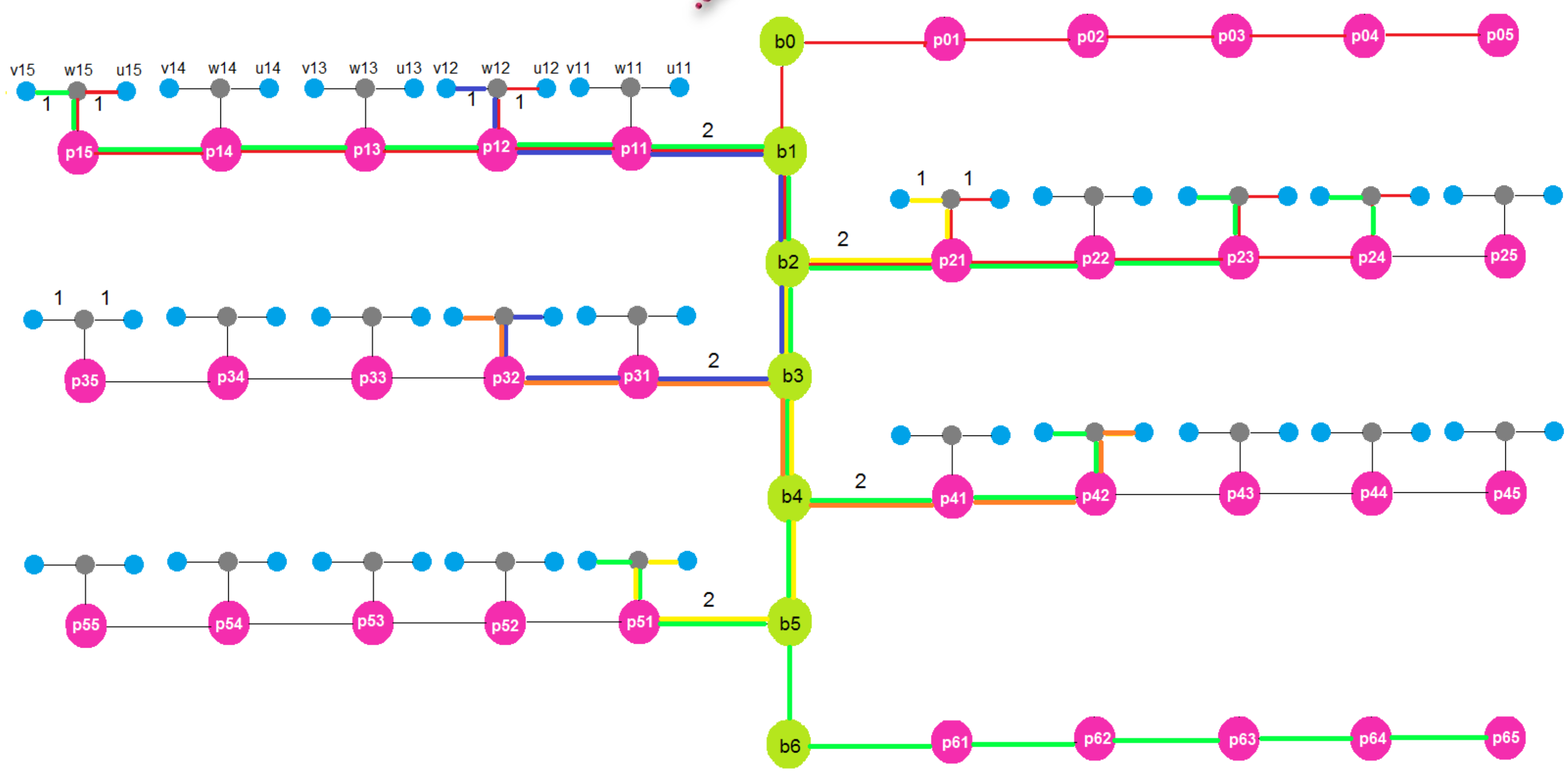
$\forall l, , 0 \leq l \leq d(i)$, from $v_{j_l,l}$ to $u_{j_{l+1},i}$



Proof of Theorem 2

All demands and capacities

W=2k



Proof of Theorem 2

Suppose that the original graph has a packing V' of size k, we will construct a CapMaxPC solution of size n^2+k.

- We select all the local demands, which gives us a solution of size n^2
- For each $i \in V'$ we increase the solution size by 1, by satisfying all the global demands associated with i.
- In each such step we use 2 new colors and remove the local demands that intersect with the global demands we selected (profit of exactly one demand)
- Fewer than 2k colors in an edge's capacity→ more than two request going through $(b_i, p_{i,1})$→ i is a common neighbor of two vertices of the packing, violating its feasibility

Proof of Theorem 2

Suppose that a solution of size $n^2 + k$ exists.

- For each edge (b_i, $p_{i,1}$) we are either satisfying two of the demands crossing it or none and furthermore that if we are satisfying two, one of them is going "left" and the other is going "right"
- The number of satisfied requests going through each edge (b_i, b_{i+1}) is constant (L) for all i. We will show that L=k.
  - ✓ Pick an arbitrary satisfied demand which uses a backbone edge and delete it from the solution (size solution and L decrease by 1)
  - ✓ Repeat this L times
  - ✓ The new solution satisfies all the local demands which are n^2, so L=k
- There are k vertices in the branch of b_0 with satisfied demands and all associated global demands are also satisfied→ a DNP in the original graph (if two of them had a common neighbor → exceeding some branch's bottleneck capacity of 2)

Proof of Theorem 2

Bi-directed trees

- For each edge (b_i, $p_{i,1}$) the capacity is set to 1
- W=k, since all global demands corresponding to a vertex are non-intersecting
- Make sure that the local demands are directed in such a way that they intersect both global demands with which they share an edge
- The rest of the arguments of the reduction go through unchanged

Structural Parameterization

Theorem 3: *$(cW, p\Delta)$-MaxPC is W[1]-hard for both undirected and bi-directed trees. The result holds even for instances where all the vertices but one have degree bounded by 3*

Proof sketch

Gadget

- Take k copies of a path on n vertices, $S_{i,j}$, $1 \le i \le k$, $1 \le j \le n$
- Take k more copies of a path on n vertices, $T_{i,j}$, $1 \le i \le k$, $1 \le j \le n$
- Add a new vertex C and connect it to all $S_{i,j}$ and $T_{i,j}$
- Set capacities of all edges to 1
- Add a demand fro $S_{i,j}$ to $T_{i,j}$

The optimal solution satisfies k paths by selecting k vertices in the S branches and their corresponding vertices in the T branches

Structural Parameterization

How do we ensure that our selection is indeed a packing in the original graph?

For all the $\binom{k}{2}$ pairs of the selection, we make copies of the gadget we constructed in theorem 2, slightly altered, and we attach them to C and then properly reroute the demands from S to T vertices through them.

The total number of vertices is $O(n^2k^2)$ and a solution of size $\binom{k}{2}n(n+4)+k$ can be achieved if the original graph has a packing of size k.

Structural Parameterization

Theorem 4: *(cW, CΔ, pt)-MaxPC and (cW, cΔ, pt)-MaxRPC are W[1]-hard for both undirected and bi-directed graphs*

The construction is similar to the one in theorem 2, but we replace the backbone with a grid, in order to achieve the constant number of colors (backbone, first and last branch) using the treewidth.

Structural Parameterization

From ETH we know it is not possible to find an independent set of size k on a n-vertex graph in time $n^{o(k)}$

The reductions in theorems 2 and 4 are linear in the parameter so no $n^{o(W)}$ *or* $n^{o(t)}$ *algorithms are possible for MaxPC*

The reductions in theorem 3 is quadratic in the parameter so no $n^{o(\sqrt{\Delta})}$ *algorithm is possible for MaxPC*

So no $n^{o(Wt\sqrt{\Delta})}$ *algorithm is possible for MaxPC*

Parameterizations involving the objective function

Theorem 5: *$(pW, p\Delta, pT)$-MaxPC is FPT for both undirected and bi-directed trees.*

Bi-directed trees:

- if all three are part of the parameter the problem is FPT
- if we drop T the problem is W[1]-hard from the results of the previous section
- if we drop any of the other two the problem is NP-hard

Undirected trees:

- if all three are part of the parameter the problem is FPT
- What happens if we drop only W from the list of parameters?

Parameterizations involving the objective function

Theorem 6: *(pT)-MaxPC is FPT on undirected trees of maximum degree 3*

Proof sketch

- PC on undirected trees can be decomposed into PC on stars
- We locate good and bad stars, pruning away the good part of the tree
- Kernelization: the new tree cannot have more than O(T) leaves → at most O(T) internal vertices of degree 3 without attached leaves
- If we remove all leaves: all vertices have degree at most 2, except the "special" vertices (degree 1 or 3) which are at most O(T)
- First endpoint dropped: one of the bad leaves
- Second endpoint dropped:
  - →guess the other endpoint among the "special" vertices or
  - →if endpoint not a "special" vertex, use the optimal greedy criterion of picking the one furthest away

Natural Parameterization of MaxPC

Theorem 7: *In any graph topology where (pB)-MaxEDP is FPT, (pB)-MaxPC is also FPT*

Proof sketch

- Obviously W<B. Randomly color all the demands using W colors. This separates the demands in W disjoint sets
- For each set solve the MaxEDP
- The union of the solutions gives us a solution to MaxPC, by using different color for each solution
- Repeat the second phase $O(\frac{W^B}{W!})$ times to create a constant probability of success

Natural Parameterization of MaxPC

Corollary 1: *(pB)-MaxPC is FPT on undirected trees and rings*

Theorem 8: *(pB)-MaxEDP is FPT on bi-directed trees*

Corollary 2: *(pB)-MaxPC is FPT on bi-directed trees*

Summary of results

Undirected			Bi-Directed		
Problem	Result	Comment	Problem	Result	Comment
(cW)-PC, $W = 3$	NP-h	(edge 3-coloring) [6]	(cW)-PC, $W = 1$	NP-h	[6]
$(p\Delta)$-PC	FPT	[6]	$(c\Delta)$-PC, $\Delta = 3$	NP-h	[6]
			$(pW, p\Delta)$-PC	FPT	[6]
$(c\Delta, ct)$-PC	NP-h	(PC on rings) [10]	$(c\Delta, ct)$-PC	NP-h	[10]
$(cW, c\Delta)$-MaxPC	P	[5]	$(cW, c\Delta)$-MaxPC	P	[5]
$(cW, c\Delta, ct)$-MaxPC	P	Theorem 1	$(cW, c\Delta, ct)$-MaxPC	P	Theorem 1
$(pW, c\Delta)$-MaxPC	W[1]-h	Theorem 2	$(pW, c\Delta)$-MaxPC	W[1]-h	Theorem 2
$(cW, p\Delta)$-MaxPC	W[1]-h	Theorem 3	$(cW, p\Delta)$-MaxPC	W[1]-h	Theorem 3
$(cW, c\Delta, pt)$-MaxPC	W[1]-h	Theorem 4	$(cW, c\Delta, pt)$-MaxPC	W[1]-h	Theorem 4
$(pW, p\Delta, pT)$-MaxPC	FPT	Theorem 5	$(pW, p\Delta, pT)$-MaxPC	FPT	Theorem 5
(pT)-MaxPC, $\Delta = 3$	FPT	Theorem 6			
(pB)-MaxPC	FPT	Corollary 1	(pB)-MaxPC	FPT	Corollary 2

All results concern trees, except those where the graphs treewidth t is included in the problem description

BIBLIOGRAPHY

- MICHAEL LAMPIS, PARAMETERIZED MAXIMUM PATH COLORING, 2011
- RODNEY G. DOWNEY & DIMITRIOS M. THILIKOS, CONFRONTING INTRACTABILITY VIA PARAMETERS, 2011
- J. FLUM & M. GROHE, PARAMETERIZED COMPLEXITY THEORY, 2006

THANK YOU!