Turán's Graph Theorem

Nils Lauermann
Computer Laboratory, University of Cambridge CB3 0FD
Nils.Lauermann@cl.cam.ac.uk

March 17, 2025

Abstract

Turán's Graph Theorem [2] states that any undirected, simple graph with n vertices that does not contain a p-clique, contains at most $\left(1 - \frac{1}{p-1}\right) \frac{n^2}{2}$ edges. The theorem is an important result in graph theory and the foundation of the field of extremal graph theory.

The formalisation follows Aigner and Ziegler's [1] presentation of Turán's initial proof [2]. Besides a direct adaptation of the textbook proof, a simplified, second proof is presented which decreases the size of the formalised proof significantly.

Contents

Acknowledgements

This formalisation was developed as part of the author's MPhil dissertation (Submitted on 6 June 2022) of the Advanced Computer Science course at the University of Cambridge. The project was supervised by Prof Lawrence C. Paulson and Dr Angeliki Koutsoukou-Argyraki.

References

[1] M. Aigner and G. M. Ziegler. *Turán's graph theorem*, pages 285–289. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018.

[2] P. Turán. On an external problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941.

theory *Turan*
  imports
    Girth-Chromatic.Ugraphs
    Random-Graph-Subgraph-Threshold.Ugraph-Lemmas
begin

1 Basic facts on graphs

lemma *wellformed-uverts-0* :
  assumes *uwellformed G* **and** *uverts G* = {}
  shows *card* (*uedges G*) = *0* **using** *assms*
  by (*metis uwellformed-def card.empty ex-in-conv zero-neq-numeral*)

lemma *finite-verts-edges* :
  assumes *uwellformed G* **and** *finite* (*uverts G*)
  shows *finite* (*uedges G*)
proof −
  have *sub-pow*: *uwellformed G* ⟹ *uedges G* ⊆ {*S*. *S* ⊆ *uverts G*}
    by (*cases G*, *auto simp add*: *uwellformed-def*)
  then have *finite* {*S*. *S* ⊆ *uverts G*} **using** *assms*
    by *auto*
  with *sub-pow assms* **show** *finite* (*uedges G*)
    using *finite-subset* **by** *blast*
qed

lemma *ugraph-max-edges* :
  assumes *uwellformed G* **and** *card* (*uverts G*) = *n* **and** *finite* (*uverts G*)
  shows *card* (*uedges G*) ≤ *n* ∗ (*n*−*1*)/*2*
  using *assms wellformed-all-edges* [*OF assms*(*1*)] *card-all-edges* [*OF assms*(*3*)] *Binomial.choose-two* [*of card*(*uverts G*)]
  by (*smt* (*verit*, *del-insts*) *all-edges-finite card-mono dbl-simps*(*3*) *dbl-simps*(*5*) *div-times-less-eq-dividend le-divide-eq-numeral1*(*1*) *le-square nat-mult-1-right numerals*(*1*) *of-nat-1 of-nat-diff of-nat-mono of-nat-mult of-nat-numeral right-diff-distrib*′)

lemma *subgraph-verts-finite* : ⟦ *finite* (*uverts G*); *subgraph G*′ *G* ⟧ ⟹ *finite* (*uverts G*′)
  using *rev-finite-subset subgraph-def* **by** *auto*

2 Cliques

In this section a straightforward definition of cliques for simple, undirected graphs is introduced. Besides fundamental facts about cliques, also more specialized lemmata are proved in subsequent subsections.

definition *uclique* :: *ugraph* ⇒ *ugraph* ⇒ *nat* ⇒ *bool* **where**
  uclique C G p ≡ *p* = *card* (*uverts C*) ∧ *subgraph C G* ∧ *C* = *complete* (*uverts C*)

lemma *clique-any-edge* :
  assumes *uclique C G p* **and** $x \in$ *uverts C* **and** $y \in$ *uverts C* **and** $x \neq y$
  shows $\{x,y\} \in$ *uedges G*
  using *assms*
  apply (*simp add*: *uclique-def complete-def all-edges-def subgraph-def*)
  by (*smt* (*verit*, *best*) *SigmaI fst-conv image-iff mem-Collect-eq mk-uedge.simps snd-conv subset-eq*)

lemma *clique-exists* : $\exists$ *C p. uclique C G p* $\wedge$ $p \leq$ *card* (*uverts G*)
  using *bex-imageD card.empty emptyE gr-implies-not0 le-neq-implies-less*
  by (*auto simp add*: *uclique-def complete-def subgraph-def all-edges-def*)

lemma *clique-exists1* :
  assumes *uverts G* $\neq$ {} **and** *finite* (*uverts G*)
  shows $\exists$ *C p. uclique C G p* $\wedge$ *0* < *p* $\wedge$ $p \leq$ *card* (*uverts G*)
proof −
  obtain *x* **where** *x*: $x \in$ *uverts G*
    using *assms*
    by *auto*
  show *?thesis*
    apply (*rule exI* [*of* - ({*x*},{})], *rule exI* [*of* - *1*])
    using *x assms*(*2*)
  by (*simp add*: *uclique-def subgraph-def complete-def all-edges-def Suc-leI assms*(*1*) *card-gt-0-iff*)
qed

lemma *clique-max-size* : *uclique C G p* $\Longrightarrow$ *finite* (*uverts G*) $\Longrightarrow$ $p \leq$ *card* (*uverts G*)
  by (*auto simp add*: *uclique-def subgraph-def Finite-Set.card-mono*)

lemma *clique-exists-gt0* :
  assumes *finite* (*uverts G*) *card* (*uverts G*) > *0*
  shows $\exists$ *C p. uclique C G p* $\wedge$ $p \leq$ *card* (*uverts G*) $\wedge$ ($\forall$ *C q. uclique C G q* $\longrightarrow$ $q \leq p$)
proof −
  have *1*: *finite* (*uverts G*) $\Longrightarrow$ *finite* {*p*. $\exists$ *C*. *uclique C G p*}
    using *clique-max-size*
    by (*smt* (*verit*, *best*) *finite-nat-set-iff-bounded-le mem-Collect-eq*)
  have *2*: $\bigwedge$*A*::*nat set*. *finite A* $\Longrightarrow$ $\exists$ *x*. $x \in A$ $\Longrightarrow$ $\exists x \in A. \forall y \in A.\ y \leq x$
    using *Max-ge Max-in* **by** *blast*
  have $\exists$ *C p. uclique C G p* $\wedge$ ($\forall$ *C q. uclique C G q* $\longrightarrow$ $q \leq p$)
    using *2* [*OF 1* [*OF* ‹*finite* (*uverts G*)›]] *clique-exists* [*of G*]
    by (*smt* (*z3*) *mem-Collect-eq*)
  then show *?thesis*
    using ‹*finite* (*uverts G*)› *clique-max-size*
    by *blast*
qed

If there exists a $(p+1)$-clique C in a graph G then we can obtain a p-clique in G by removing an arbitrary vertex from C

lemma *clique-size-jumpfree* :
assumes *finite* (*uverts* G) **and** *uwellformed* G
and *uclique* C G (p+*1*)
shows $\exists$ C'. *uclique* C' G p
proof −
have *card*(*uverts* G) > p
using *assms* **by** (*simp add*: *uclique-def subgraph-def card-mono less-eq-Suc-le*)
obtain x **where** x: $x \in$ *uverts* C
using *assms* **by** (*fastforce simp add*: *uclique-def*)
have *mk-uedge* ‘ {*uv* ∈ *uverts* C × *uverts* C. *fst uv* ≠ *snd uv*} − {A ∈ *uedges* C. $x \in A$} =
mk-uedge ‘ {*uv* ∈ (*uverts* C − {x}) × (*uverts* C − {x}). *fst uv* ≠ *snd uv*}
proof −
have ⋀y. y ∈ *mk-uedge* ‘ {*uv* ∈ *uverts* C × *uverts* C. *fst uv* ≠ *snd uv*} − {A ∈ *uedges* C. $x \in A$} ⟹
y ∈ *mk-uedge* ‘ {*uv* ∈ (*uverts* C − {x}) × (*uverts* C − {x}). *fst uv* ≠ *snd uv*}
using *assms*(*3*)
apply (*simp add*: *uclique-def complete-def all-edges-def*)
by (*smt* (*z3*) *DiffI SigmaE SigmaI image-iff insertCI mem-Collect-eq mk-uedge.simps singleton-iff snd-conv*)
moreover have ⋀y. y ∈ *mk-uedge* ‘ {*uv* ∈ (*uverts* C − {x}) × (*uverts* C − {x}). *fst uv* ≠ *snd uv*}
⟹ y ∈ *mk-uedge* ‘ {*uv* ∈ *uverts* C × *uverts* C. *fst uv* ≠ *snd uv*} − {A ∈ *uedges* C. $x \in A$}
apply (*simp add*: *uclique-def complete-def all-edges-def*)
by (*smt* (*z3*) *DiffE SigmaE SigmaI image-iff insert-iff mem-Collect-eq mk-uedge.simps singleton-iff*)
ultimately show *?thesis*
by *blast*
qed
then have *1*: (*uverts* C − {x}, *uedges* C − {A ∈ *uedges* C. $x \in A$}) = *Ugraph-Lemmas.complete* (*uverts* C − {x})
using *assms*(*3*)
apply (*simp add*: *uclique-def complete-def all-edges-def*)
by (*metis* (*no-types*, *lifting*) *snd-eqD*)
show *?thesis*
apply (*rule exI* [*of - C −− x*])
using *assms x*
apply (*simp add*: *uclique-def remove-vertex-def subgraph-def*)
apply (*simp add*: *1*)
by (*auto simp add*: *complete-def all-edges-def*)
qed

The next lemma generalises the lemma *clique-size-jumpfree* to a proof of the existence of a clique of any size smaller than the size of the original clique.

lemma *clique-size-decr* :
assumes *finite* (*uverts* G) **and** *uwellformed* G

and *uclique C G p*
shows $q \leq p \Longrightarrow \exists\, C.$ *uclique C G q* **using** *assms*
proof (*induction q rule*: *measure-induct* [*of* $\lambda x.\ p - x$])
case (*1 x*)
then show *?case*
proof (*cases* $x = p$)
case *True*
then show *?thesis*
using ‹*uclique C G p*›
by *blast*
next
case *False*
with *1*(*2*) **have** $x < p$
by *auto*
from ‹$x < p$› **have** $p - Suc\ x < p - x$
by *auto*
then show *?thesis*
using *1*(*1*) *assms*(*1*,*2*,*3*) ‹$x < p$›
using *clique-size-jumpfree* [*OF* ‹*finite* (*uverts G*)› ‹*uwellformed G*› -]
by (*metis 1.prems*(*4*) *add.commute linorder-not-le not-less-eq plus-1-eq-Suc*)
qed
qed

With this lemma we can easily derive by contradiction that if there is no p-clique then there cannot exist a clique of a size greater than p

corollary *clique-size-neg-max* :
assumes *finite* (*uverts G*) **and** *uwellformed G*
and $\neg(\exists\, C.$ *uclique C G p*)
shows $\forall\, C\ q.$ *uclique C G q* $\longrightarrow q < p$
proof (*rule ccontr*)
assume *1*: $\neg$ ($\forall\, C\ q.$ *uclique C G q* $\longrightarrow q < p$)
show *False*
proof −
obtain *C q* **where** *C*: *uclique C G q*
and *q*: $q \geq p$
using *1 linorder-not-less*
by *blast*
show *?thesis*
using *assms*(*3*) *q clique-size-decr* [*OF* ‹*finite* (*uverts G*)› ‹*uwellformed G*› *C*
]
using *order-less-imp-le* **by** *blast*
qed
qed

corollary *clique-complete* :
assumes *finite V* **and** $x \leq$ *card V*
shows $\exists\, C.$ *uclique C* (*complete V*) *x*
proof −
have *uclique* (*complete V*) (*complete V*) (*card V*)

by (*simp add*: *uclique-def complete-def subgraph-def*)
then show *?thesis*
using *clique-size-decr* [*OF - complete-wellformed* [*of V*] - *assms*(*2*)] *assms*(*1*)
by (*simp add*: *complete-def*)
qed

lemma *subgraph-clique* :
assumes *uwellformed G subgraph C G C* = *complete* (*uverts C*)
shows {*e* ∈ *uedges G*. *e* ⊆ *uverts C*} = *uedges C*
proof −
from *assms complete-wellformed* [*of uverts C*] **have** *uedges C* ⊆ {*e* ∈ *uedges G*. *e* ⊆ *uverts C*}
by (*auto simp add*: *subgraph-def uwellformed-def*)
moreover from *assms*(*1*) *complete-wellformed* [*of uverts C*] **have** {*e* ∈ *uedges G*. *e* ⊆ *uverts C*} ⊆ *uedges C*
apply (*simp add*: *subgraph-def uwellformed-def complete-def card-2-iff all-edges-def*)
using *assms*(*3*)[*unfolded complete-def all-edges-def*] *in-mk-uedge-img*
by (*smt* (*verit*, *ccfv-threshold*) *SigmaI fst-conv insert-subset mem-Collect-eq snd-conv subsetI*)
ultimately show *?thesis*
by *auto*
qed

Next, we prove that in a graph G with a p-clique C and some vertex v outside of this clique, there exists a $(p + 1)$-clique in G if v is connected to all nodes in C. The next lemma is an abstracted version that does not explicitly mention cliques: If a vertex n has as many edges to a set of nodes N as there are nodes in N then n is connected to all vertices in N.

lemma *card-edges-nodes-all-edges* :
fixes *G* :: *ugraph* **and** *N* :: *nat set* **and** *E* :: *nat set set* **and** *n* :: *nat*
assumes *uwellformed G*
and *finite N*
and *N* ⊆ *uverts G* **and** *E* ⊆ *uedges G*
and *n* ∈ *uverts G* **and** *n* ∉ *N*
and ∀ *e* ∈ *E*. ∃ *x* ∈ *N*. {*n*,*x*} = *e*
and *card E* = *card N*
shows ∀ *x* ∈ *N*. {*n*,*x*} ∈ *E*
proof (*rule ccontr*)
assume ¬(∀ *x* ∈ *N*. {*n*,*x*} ∈ *E*)
show *False*
proof −
obtain *x* **where** *x*: *x* ∈ *N* **and** *e*: {*n*,*x*} ∉ *E*
using ‹¬(∀ *x* ∈ *N*. {*n*,*x*} ∈ *E*)›
by *auto*
have *E* ⊆ (λ*y*. {*n*,*y*}) ‘ (*N* − {*x*})
using *Set.image-diff-subset* ‹∀ *e* ∈ *E*. ∃ *x* ∈ *N*. {*n*,*x*} = *e*› *x e*
by *auto*
then show *?thesis*
using ‹*finite N*› ‹*card E* = *card N*› *x*

using *surj-card-le* [*of N* − {*x*} *E* (λ*y*. {*n*,*y*})]
by (*simp*, *metis card-gt-0-iff diff-less emptyE lessI linorder-not-le*)
qed
qed

2.1 Partitioning edges along a clique

Turán's proof partitions the edges of a graph into three partitions for a $(p-1)$-clique C: All edges within C, all edges outside of C, and all edges between a vertex in C and a vertex not in C.

We prove a generalized lemma that partitions the edges along some arbitrary set of vertices which does not necessarily need to induce a clique. Furthermore, in Turán's graph theorem we only argue about the cardinality of the partitions so that we restrict this proof to showing that the sum of the cardinalities of the partitions is equal to number of all edges.

lemma *graph-partition-edges-card* :
assumes *finite* (*uverts G*) **and** *uwellformed G* **and** *A* ⊆ (*uverts G*)
shows *card* (*uedges G*) = *card* {*e* ∈ *uedges G*. *e* ⊆ *A*} + *card* {*e* ∈ *uedges G*. *e* ⊆ *uverts G* − *A*} + *card* {*e* ∈ *uedges G*. *e* ∩ *A* ≠ {} ∧ *e* ∩ (*uverts G* − *A*) ≠ {}}
using *assms*
proof −
have *uedges G* = {*e* ∈ *uedges G*. *e* ⊆ *A*} ∪ {*e* ∈ *uedges G*. *e* ⊆ (*uverts G*) − *A*} ∪ {*e* ∈ *uedges G*. *e* ∩ *A* ≠ {} ∧ *e* ∩ ((*uverts G*) − *A*) ≠ {}}
using *assms uwellformed-def*
by *blast*
moreover have {*e* ∈ *uedges G*. *e* ⊆ *A*} ∩ {*e* ∈ *uedges G*. *e* ⊆ *uverts G* − *A*} = {}
using *assms uwellformed-def*
by (*smt* (*verit*, *ccfv-SIG*) *Diff-disjoint Int-subset-iff card.empty disjoint-iff mem-Collect-eq nat.simps*(*3*) *nat-1-add-1 plus-1-eq-Suc prod.sel*(*2*) *subset-empty*)
moreover have ({*e* ∈ *uedges G*. *e* ⊆ *A*} ∪ {*e* ∈ *uedges G*. *e* ⊆ *uverts G* − *A*}) ∩ {*e* ∈ *uedges G*. *e* ∩ *A* ≠ {} ∧ *e* ∩ (*uverts G* − *A*) ≠ {}} = {}
by *blast*
moreover have *finite* {*e* ∈ *uedges G*. *e* ⊆ *A*} **using** *assms*
by (*simp add*: *finite-subset*)
moreover have *finite* {*e* ∈ *uedges G*. *e* ⊆ *uverts G* − *A*} **using** *assms*
by (*simp add*: *finite-subset*)
moreover have *finite* {*e* ∈ *uedges G*. *e* ∩ *A* ≠ {} ∧ *e* ∩ (*uverts G* − *A*) ≠ {}}
using *assms finite-verts-edges*
by *auto*
ultimately show *?thesis*
using *assms Finite-Set.card-Un-disjoint*
by (*smt* (*verit*, *best*) *finite-UnI*)
qed

Now, we turn to the problem of calculating the cardinalities of these partitions when they are induced by the biggest clique in the graph.

First, we consider the number of edges in a p-clique.

lemma *clique-edges-inside* :
assumes *G1*: *uwellformed G* **and** *G2*: *finite* (*uverts G*)
and *p*: $p \leq$ *card* (*uverts G*) **and** *n*: $n =$ *card*(*uverts G*)
and *C*: *uclique C G p*
shows *card* {*e* ∈ *uedges G*. *e* ⊆ *uverts C*} = *p* ∗ (*p*−*1*) / *2*
proof −
have *2 dvd* (*card* (*uverts C*) ∗ (*p* − *1*))
using *C uclique-def*
by *auto*
have *2* = *real 2*
by *simp*
then show *?thesis*
using *C uclique-def* [*of C G p*] *complete-def* [*of uverts C*]
using *subgraph-clique* [*OF G1*, *of C*] *subgraph-verts-finite* [*OF assms*(*2*), *of C*]
using *Real.real-of-nat-div* [*OF* ‹*2 dvd* (*card* (*uverts C*) ∗ (*p* − *1*))›] *Binomial.choose-two* [*of card* (*uverts G*)]
by (*smt* (*verit*, *del-insts*) *One-nat-def approximation-preproc-nat*(*5*) *card-all-edges diff-self-eq-0 eq-imp-le left-diff-distrib′ left-diff-distrib′ linorder-not-less mult-le-mono2 choose-two not-gr0 not-less-eq-eq of-nat-1 of-nat-diff snd-eqD*)
qed

Next, we turn to the number of edges that connect a node inside of the biggest clique with a node outside of said clique. For that we start by calculating a bound for the number of edges from one single node outside of the clique into the clique.

lemma *clique-edges-inside-to-node-outside* :
assumes *uwellformed G* **and** *finite* (*uverts G*)
assumes $0 < p$ **and** $p \leq$ *card* (*uverts G*)
assumes *uclique C G p* **and** (∀ *C p′*. *uclique C G p′* ⟶ *p′* ≤ *p*)
assumes *y*: *y* ∈ *uverts G* − *uverts C*
shows *card* {{*x*,*y*}| *x*. *x* ∈ *uverts C* ∧ {*x*,*y*} ∈ *uedges G*} ≤ *p* − *1*
proof (*rule ccontr*)

For effective proof automation we use a local function definition to compute this set of edges into the clique from any node *y*:

define *S* **where** *S* ≡ λ*y*. {{*x*,*y*}| *x*. *x* ∈ *uverts C* ∧ {*x*,*y*} ∈ *uedges G*}
assume ¬ *card* {{*x*, *y*} |*x*. *x* ∈ *uverts C* ∧ {*x*, *y*} ∈ *uedges G*} ≤ *p* − *1*
then have *Sy*: *card* (*S y*) > *p* − *1*
using *S-def y* **by** *auto*
have *uclique* ({*y*} ∪ (*uverts C*),*S y* ∪ *uedges C*) *G* (*Suc p*)
proof −
have *card* ({*y*} ∪ *uverts C*) = *Suc p*
using *assms*(*3*,*5*,*7*) *uclique-def*
by (*metis DiffD2 card-gt-0-iff card-insert-disjoint insert-is-Un*)
moreover have *subgraph* ({*y*} ∪ *uverts C*, (*S y*) ∪ *uedges C*) *G*
using *assms*(*5*,*7*)
by (*auto simp add*: *uclique-def subgraph-def S-def*)
moreover have ({*y*} ∪ (*uverts C*),(*S y*) ∪ *uedges C*) = *complete* ({*y*} ∪ (*uverts C*))

proof −
have $(S\ y) \cup uedges\ C \subseteq all\text{-}edges\ (\{y\} \cup (uverts\ C))$
using *y assms(5) S-def all-edges-def uclique-def complete-def*
by (*simp, smt (z3) SigmaE SigmaI fst-conv image-iff in-mk-uedge-img insertCI mem-Collect-eq snd-conv subsetI*)
moreover have $all\text{-}edges\ (\{y\} \cup (uverts\ C)) \subseteq (S\ y) \cup uedges\ C$
proof −
have $\forall x \in uverts\ C.\ \{y, x\} \in S\ y$
proof −
have $card\ (S\ y) = card\ (uverts\ C)$
using *Sy assms(2,3,5,7) S-def uclique-def card-gt-0-iff*
using *Finite-Set.surj-card-le* [*of uverts C S y* $\lambda x.\ \{x, y\}$]
by (*smt (verit, del-insts) Suc-leI Suc-pred′ image-iff le-antisym mem-Collect-eq subsetI*)
then show *?thesis*
using *card-edges-nodes-all-edges* [*OF assms(1), of uverts C S y y*] *assms(1,2,5,7) S-def uclique-def*
by (*smt (verit, ccfv-threshold) DiffE insert-commute mem-Collect-eq subgraph-def subgraph-verts-finite subsetI*)
qed
then show *?thesis*
using *assms(5) all-edges-def S-def uclique-def complete-def mk-uedge.simps in-mk-uedge-img*
by (*smt (z3) insert-commute SigmaI fst-conv mem-Collect-eq snd-conv SigmaE UnCI image-iff insert-iff insert-is-Un subsetI*)
qed
ultimately show *?thesis*
by (*auto simp add: complete-def*)
qed
ultimately show *?thesis*
by (*simp add: uclique-def complete-def*)
qed
then show *False*
using *assms(6)*
by *fastforce*
qed

Now, that we have this upper bound for the number of edges from a single vertex into the largest clique we can calculate the upper bound for all such vertices and edges:

lemma *clique-edges-inside-to-outside* :
assumes *G1*: *uwellformed G* **and** *G2*: *finite (uverts G)*
and *p0*: $0 < p$ **and** *pn*: $p \leq card\ (uverts\ G)$ **and** $card(uverts\ G) = n$
and *C*: *uclique C G p* **and** *C-max*: $(\forall C\ p'.\ uclique\ C\ G\ p' \longrightarrow p' \leq p)$
shows $card\ \{e \in uedges\ G.\ e \cap uverts\ C \neq \{\} \wedge e \cap (uverts\ G - uverts\ C) \neq \{\}\} \leq (p - 1) * (n - p)$
proof −
define *S* **where** $S \equiv \lambda y.\ \{\{x,y\} |\ x.\ x \in uverts\ C \wedge \{x,y\} \in uedges\ G\}$
have $card\ (uverts\ G - uverts\ C) = n - p$

using *pn C* ‹*card*(*uverts G*) = *n*› *G2*
apply (*simp add*: *uclique-def*)
by (*meson card-Diff-subset subgraph-def subgraph-verts-finite*)
moreover have {*e* ∈ *uedges G*. *e* ∩ *uverts C* ≠ {} ∧ *e* ∩ (*uverts G* − *uverts C*) ≠ {}} = {{*x*,*y*}| *x y*. *x* ∈ *uverts C* ∧ *y* ∈ (*uverts G* − *uverts C*) ∧ {*x*,*y*} ∈ *uedges G*}
proof −
have *e* ∈ {*e* ∈ *uedges G*. *e* ∩ *uverts C* ≠ {} ∧ *e* ∩ (*uverts G* − *uverts C*) ≠ {}}
⟹ ∃ *x y*. *e* = {*x*,*y*} ∧ *x* ∈ *uverts C* ∧ *y* ∈ *uverts G* − *uverts C* **for** *e*
using *G1*
apply (*simp add*: *uwellformed-def*)
by (*smt* (*z3*) *DiffD2 card-2-iff disjoint-iff-not-equal insert-Diff insert-Diff-if insert-iff*)
then show *?thesis*
by *auto*
qed
moreover have *card* {{*x*,*y*}| *x y*. *x* ∈ *uverts C* ∧ *y* ∈ (*uverts G* − *uverts C*) ∧ {*x*,*y*} ∈ *uedges G*} ≤ *card* (*uverts G* − *uverts C*) ∗ (*p*−*1*)
proof −
have *card* {{*x*,*y*}| *x y*. *x* ∈ *uverts C* ∧ *y* ∈ (*uverts G* − *uverts C*) ∧ {*x*,*y*} ∈ *uedges G*}
≤ (∑ *y* ∈ (*uverts G* − *uverts C*). *card* (*S y*))
proof −
have *finite* (*uverts G* − *uverts C*)
using ‹*finite* (*uverts G*)› **by** *auto*
have {{*x*,*y*}| *x y*. *x* ∈ *uverts C* ∧ *y* ∈ (*uverts G* − *uverts C*) ∧ {*x*,*y*} ∈ *uedges G*}
= (⋃ *y* ∈ (*uverts G* − *uverts C*). {{*x*,*y*}| *x*. *x* ∈ *uverts C* ∧ {*x*,*y*} ∈ *uedges G*})
by *auto*
then show *?thesis*
using *Groups-Big.card-UN-le* [*OF* ‹*finite* (*uverts G* − *uverts C*)›,
of λ*y*. {{*x*, *y*} |*x*. *x* ∈ *uverts C* ∧ {*x*, *y*} ∈ *uedges G*}]
using *S-def*
by *auto*
qed
moreover have (∑ *y*∈*uverts G* − *uverts C*. *card* (*S y*)) ≤ *card* (*uverts G* − *uverts C*) ∗ (*p*−*1*)
proof −
have *card* (*S y*) ≤ *p* − *1* **if** *y*: *y* ∈ *uverts G* − *uverts C* **for** *y*
using *clique-edges-inside-to-node-outside* [*OF assms*(*1*,*2*,*3*,*4*) *C C-max y*] *S-def y*
by *simp*
then show *?thesis*
by (*metis id-apply of-nat-eq-id sum-bounded-above*)
qed
ultimately show *?thesis*
using *order-trans*

```
    by blast
  qed
  ultimately show ?thesis
    by (smt (verit, ccfv-SIG) mult.commute)
qed
```

Lastly, we need to argue about the number of edges which are located entirely outside of the greatest clique. Note that this is in the inductive step case in the overarching proof of Turán's graph theorem. That is why we have access to the inductive hypothesis as an assumption in the following lemma:

```
lemma clique-edges-outside :
  assumes uwellformed G and finite (uverts G)
    and p2: 2 ≤ p and pn: p ≤ card (uverts G) and n: n = card(uverts G)
    and C: uclique C G (p−1) and C-max: (∀ C q. uclique C G q ⟶ q ≤ p−1)
     and IH: ⋀G y. y < n ⟹ finite (uverts G) ⟹ uwellformed G ⟹ ∀ C p′.
uclique C G p′ ⟶ p′ < p
               ⟹ 2 ≤ p ⟹ card (uverts G) = y ⟹ real (card (uedges G)) ≤ (1
− 1 / real (p − 1)) ∗ real (y^2) / 2
  shows card {e ∈ uedges G. e ⊆ uverts G − uverts C} ≤ (1 − 1 / (p−1)) ∗ (n
− p + 1) ^ 2 / 2
proof −
  have n − card (uverts C) < n
    using C pn p2 n
   by (metis Suc-pred′ diff-less less-2-cases-iff linorder-not-less not-gr0 uclique-def)
  have GC1: finite (uverts (uverts G − uverts C, {e ∈ uedges G. e ⊆ uverts G −
uverts C}))
    using assms(2)
    by simp
  have GC2: uwellformed (uverts G − uverts C, {e ∈ uedges G. e ⊆ uverts G −
uverts C})
    using assms(1)
    by (auto simp add: uwellformed-def)
  have GC3: ∀ C′ p′. uclique C′ (uverts G − uverts C, {e ∈ uedges G. e ⊆ uverts
G − uverts C}) p′ ⟶ p′ < p
  proof (rule ccontr)
   assume ¬(∀ C′ p′. uclique C′ (uverts G − uverts C, {e ∈ uedges G. e ⊆ uverts
G − uverts C}) p′ ⟶ p′ < p)
    then obtain C′ p′ where C′: uclique C′ (uverts G − uverts C, {e ∈ uedges
G. e ⊆ uverts G − uverts C}) p′ and p′: p′ ≥ p
      by auto
    then have uclique C′ G p′
      using uclique-def subgraph-def
      by auto
    then show False
      using p′ p2 C-max
      by fastforce
  qed
  have GC4: card (uverts (uverts G − uverts C,{e ∈ uedges G. e ⊆ uverts G −
uverts C})) = n − card (uverts C)
```

using *C n assms*(*2*) *uclique-def subgraph-def*
by (*simp*, *meson card-Diff-subset infinite-super*)
show *?thesis*
using *C GC3 IH* [*OF* ‹*n* − *card* (*uverts C*) < *n*› *GC1 GC2 GC3* ‹*2* ≤ *p*› *GC4*] *assms*(*2*) *n uclique-def*
by (*simp*, *smt* (*verit*, *best*) *C One-nat-def Suc-1 Suc-leD clique-max-size of-nat-1 of-nat-diff p2*)
qed

2.2 Extending the size of the biggest clique

In this section, we want to prove that we can add edges to a graph so that we augment the biggest clique to some greater clique with a specific number of vertices. For that, we need the following lemma: When too many edges have been added to a graph so that there exists a $(p+1)$-clique then we can remove at least one of the added edges while also retaining a p-clique

lemma *clique-union-size-decr* :
assumes *finite* (*uverts G*) **and** *uwellformed* (*uverts G*, *uedges G* ∪ *E*)
and *uclique C* (*uverts G*, *uedges G* ∪ *E*) (*p*+*1*)
and *card E* ≥ *1*
shows ∃ *C′ E′*. *card E′* < *card E* ∧ *uclique C′* (*uverts G*, *uedges G* ∪ *E′*) *p* ∧ *uwellformed* (*uverts G*, *uedges G* ∪ *E′*)
proof (*cases* ∃ *x* ∈ *uverts C*. ∃ *e* ∈ *E*. *x* ∈ *e*)
case *True*
then obtain *x* **where** *x1*: *x* ∈ *uverts C* **and** *x2*: ∃ *e* ∈ *E*. *x* ∈ *e*
by *auto*
show *?thesis*
proof (*rule exI* [*of* - *C* −− *x*], *rule exI* [*of* - {*e* ∈ *E*. *x* ∉ *e*}])
have *card* {*e* ∈ *E*. *x* ∉ *e*} < *card E*
using *x2 assms*(*4*)
by (*smt* (*verit*) *One-nat-def card.infinite diff-is-0-eq mem-Collect-eq minus-nat.diff-0 not-less-eq psubset-card-mono psubset-eq subset-eq*)
moreover have *uclique* (*C* −− *x*) (*uverts G*, *uedges G* ∪ {*e* ∈ *E*. *x* ∉ *e*}) *p*
proof −
have *p* = *card* (*uverts* (*C* −− *x*))
using *x1 assms*(*3*)
by (*auto simp add*: *uclique-def remove-vertex-def*)
moreover have *subgraph* (*C* −− *x*) (*uverts G*, *uedges G* ∪ {*e* ∈ *E*. *x* ∉ *e*})
using *assms*(*3*)
by (*auto simp add*: *uclique-def subgraph-def remove-vertex-def*)
moreover have *C* −− *x* = *Ugraph-Lemmas.complete* (*uverts* (*C* −− *x*))
proof −
have *1*: ⋀*y*. *y* ∈ *mk-uedge* ‘ {*uv* ∈ *uverts C* × *uverts C*. *fst uv* ≠ *snd uv*} − {*A* ∈ *uedges C*. *x* ∈ *A*} ⟹
y ∈ *mk-uedge* ‘ {*uv* ∈ (*uverts C* − {*x*}) × (*uverts C* − {*x*}). *fst uv* ≠ *snd uv*}
by (*smt* (*z3*) *DiffE DiffI SigmaE SigmaI Ugraph-Lemmas.complete-def all-edges-def assms*(*3*) *empty-iff image-iff insert-iff mem-Collect-eq mk-uedge.simps*

snd-conv uclique-def)
have *2*: ⋀*y*. *y* ∈ *mk-uedge* ‘ {*uv* ∈ (*uverts C* − {*x*}) × (*uverts C* − {*x*}). *fst uv* ≠ *snd uv*} ⟹
y ∈ *mk-uedge* ‘ {*uv* ∈ *uverts C* × *uverts C*. *fst uv* ≠ *snd uv*} − {*A* ∈ *uedges C*. *x* ∈ *A*}
by (*smt* (*z3*) *DiffE DiffI SigmaE SigmaI image-iff insert-iff mem-Collect-eq mk-uedge.simps singleton-iff*)
show *?thesis*
using *assms*(*3*)
apply (*simp add*: *remove-vertex-def complete-def all-edges-def uclique-def*)
using *1 2*
by (*smt* (*verit*, *ccfv-SIG*) *split-pairs subset-antisym subset-eq*)
qed
ultimately show *?thesis*
by (*simp add*: *uclique-def*)
qed
moreover have *uwellformed* (*uverts G*, *uedges G* ∪ {*e* ∈ *E*. *x* ∉ *e*})
using *assms*(*2*)
by (*auto simp add*: *uwellformed-def*)
ultimately show *card* {*e* ∈ *E*. *x* ∉ *e*} < *card E* ∧
uclique (*C* −− *x*) (*uverts G*, *uedges G* ∪ {*e* ∈ *E*. *x* ∉ *e*}) *p* ∧
uwellformed (*uverts G*, *uedges G* ∪ {*e* ∈ *E*. *x* ∉ *e*})
by *auto*
qed
next
case *False*
then have ⋀*x*. *x* ∈ *uedges C* ⟹ *x* ∉ *E*
using *assms*(*2*)
by (*metis assms*(*3*) *card-2-iff′ complete-wellformed uclique-def uwellformed-def*)
then have *uclique C G* (*p*+*1*)
using *assms*(*3*)
by (*auto simp add*: *uclique-def subgraph-def uwellformed-def*)
show *?thesis*
using *assms*(*2*,*4*) *clique-size-jumpfree* [*OF assms*(*1*) - ‹*uclique C G* (*p*+*1*)›]
apply (*simp add*: *uwellformed-def*)
by (*metis Suc-le-eq UnCI Un-empty-right card.empty prod.exhaust-sel*)
qed

We use this preceding lemma to prove the next result. In this lemma we assume that we have added too many edges. The goal is then to remove some of the new edges appropriately so that it is indeed guaranteed that there is no bigger clique.

Two proofs of this lemma will be described in the following. Both fundamentally come down to the same core idea: In essence, both proofs apply the well-ordering principle. In the first proof we do so immediately by obtaining the minimum of a set:

lemma *clique-union-make-greatest* :
fixes *p n* :: *nat*

```
  assumes finite (uverts G) and uwellformed G
    and uwellformed (uverts G, uedges G ∪ E) and card(uverts G) ≥ p
    and uclique C (uverts G, uedges G ∪ E) p
    and ∀ C' q'. uclique C' G q' ⟶ q' < p and 1 ≤ card E
  shows ∃ C' E'. uwellformed (uverts G, uedges G ∪ E')
        ∧ (uclique C' (uverts G, uedges G ∪ E') p)
        ∧ (∀ C'' q'. uclique C'' (uverts G, uedges G ∪ E') q' ⟶ q' ≤ p)
  using assms
proof (induction card E arbitrary: C E rule: less-induct)
  case (less E)
  then show ?case
  proof (cases ∃ A. uclique A (uverts G, uedges G ∪ E) (p+1))
    case True
    then obtain A where A: uclique A (uverts G, uedges G ∪ E) (p+1)
      by auto
    obtain C' E' where E'1: card E' < card E
      and E'2: uclique C' (uverts G, uedges G ∪ E') p
      and E'3: uwellformed (uverts G, uedges G ∪ E')
      and E'4: 1 ≤ card E'
      using less(7)
      using clique-union-size-decr [OF assms(1) ‹uwellformed (uverts G, uedges G
∪ E)› A less(8)]
        by (metis One-nat-def Suc-le-eq Un-empty-right card-gt-0-iff finite-Un fi-
nite-verts-edges fst-conv less.prems(1) less-not-refl prod.collapse snd-conv)
    show ?thesis
      using less(1) [OF E'1 assms(1,2) E'3 less(5) E'2 less(7) E'4]
      using E'1 less(8)
      by (meson less-or-eq-imp-le order-le-less-trans)
  next
    case False
    show ?thesis
      apply (rule exI [of - C], rule exI [of - E])
      using clique-size-neg-max [OF - less(4) False]
      using less(2,4,6)
      by fastforce
  qed
qed
```

In this second, alternative proof the well-ordering principle is used through complete induction.

```
lemma clique-union-make-greatest-alt :
  fixes p n :: nat
  assumes finite (uverts G) and uwellformed G
    and uwellformed (uverts G, uedges G ∪ E) and card(uverts G) ≥ p
    and uclique C (uverts G, uedges G ∪ E) p
    and ∀ C' q'. uclique C' G q' ⟶ q' < p and 1 ≤ card E
  shows ∃ C' E'. uwellformed (uverts G, uedges G ∪ E')
        ∧ (uclique C' (uverts G, uedges G ∪ E') p)
        ∧ (∀ C'' q'. uclique C'' (uverts G, uedges G ∪ E') q' ⟶ q' ≤ p)
```

proof −
define *P* **where** $P \equiv \lambda E.$ *uwellformed* (*uverts G*, *uedges G* ∪ *E*) ∧ (∃ *C*. *uclique C* (*uverts G*, *uedges G* ∪ *E*) *p*)
have *finite* {*y*. ∃ *E*. *P E* ∧ *card E* = *y*}
proof −
have ⋀*E*. *P E* ⟹ *E* ⊆ *Pow* (*uverts G*)
by (*auto simp add*: *P-def uwellformed-def*)
then have *finite* {*E*. *P E*}
using *assms*(*1*)
by (*metis Collect-mono Pow-def finite-Pow-iff rev-finite-subset*)
then show *?thesis*
by *simp*
qed
obtain *F* **where** *F1*: *P F*
and *F2*: *card F* = *Min* {*y*. ∃ *E*. *P E* ∧ *card E* = *y*}
and *F3*: *card F* > *0*
using *assms*(*1*,*3*,*4*,*5*,*6*) *Min-in* ‹*finite* {*y*. ∃ *E*. *P E* ∧ *card E* = *y*}› *P-def CollectD Collect-empty-eq*
by (*smt* (*verit*, *ccfv-threshold*) *Un-empty-right card-gt-0-iff finite-Un finite-verts-edges fst-conv le-refl linorder-not-le prod.collapse snd-conv*)
have *p* > *0*
using *assms*(*6*) *clique-exists bot-nat-0.not-eq-extremum*
by *blast*
then show *?thesis*
proof (*cases* ∃ *C*. *uclique C* (*uverts G*, *uedges G* ∪ *F*) (*p* + *1*))
case *True*
then obtain *F′* **where** *F′1* : *P F′* **and** *F′2*: *card F′* < *card F*
using *F1 F2 F3 clique-union-size-decr* [*OF assms*(*1*), *of F - p*] *P-def*
by (*smt* (*verit*) *One-nat-def Suc-eq-plus1 Suc-leI add-2-eq-Suc′ assms*(*1*) *clique-size-jumpfree fst-conv*)
then show *?thesis*
using *F2* ‹*finite* {*y*. ∃ *F*. *P F* ∧ *card F* = *y*}› *Min-gr-iff*
by *fastforce*
next
case *False*
then show *?thesis*
using *clique-size-neg-max* [*OF* - - *False*]
using *assms*(*1*) *F1 P-def*
by (*smt* (*verit*, *ccfv-SIG*) *Suc-eq-plus1 Suc-leI fst-conv linorder-not-le*)
qed
qed

Finally, with this lemma we can turn to this section's main challenge of increasing the greatest clique size of a graph by adding edges.

lemma *clique-add-edges-max* :
fixes *p* :: *nat*
assumes *finite* (*uverts G*)
and *uwellformed G* **and** *card*(*uverts G*) > *p*
and ∃ *C*. *uclique C G p* **and** (∀ *C q′*. *uclique C G q′* ⟶ *q′* ≤ *p*)

and $q \leq$ *card*(*uverts G*) **and** $p \leq q$
shows $\exists E$. *uwellformed* (*uverts G*, *uedges G* $\cup$ *E*) $\wedge$ ($\exists C$. *uclique C* (*uverts G*, *uedges G* $\cup$ *E*) *q*)
$\wedge$ ($\forall C\ q'$. *uclique C* (*uverts G*, *uedges G* $\cup$ *E*) $q' \longrightarrow q' \leq q$)
proof (*cases p* $<$ *q*)
case *True*
then show *?thesis*
proof –
have $\exists E$. *uwellformed* (*uverts G*, *uedges G* $\cup$ *E*) $\wedge$ ($\exists C$. *uclique C* (*uverts G*, *uedges G* $\cup$ *E*) *q*) $\wedge$ *card E* $\geq$ *1*
apply (*rule exI* [*of - all-edges* (*uverts G*)])
using *Set.Un-absorb1* [*OF wellformed-all-edges* [*OF assms*(*2*)]]
using *complete-wellformed* [*of uverts G*] *clique-complete* [*OF assms*(*1*,*6*)]
using *all-edges-def assms*(*1*,*5*)
apply (*simp add*: *complete-def*)
by (*metis Suc-leI True Un-empty-right all-edges-finite card-gt-0-iff linorder-not-less prod.collapse*)
then obtain *E C* **where** *E1*: *uwellformed* (*uverts G*, *uedges G* $\cup$ *E*)
and *E2*: *uclique C* (*uverts G*, *uedges G* $\cup$ *E*) *q*
and *E3*: *card E* $\geq$ *1*
by *auto*
show *?thesis*
using *clique-union-make-greatest* [*OF assms*(*1*,*2*) *E1 assms*(*6*) *E2 - E3*] *assms*(*5*) *True*
using *order-le-less-trans*
by *blast*
qed
next
case *False*
show *?thesis*
apply (*rule exI* [*of -* {}])
using *False assms*(*2*,*4*,*5*,*7*)
by *simp*
qed

3 Properties of the upper edge bound

In this section we prove results about the upper edge bound in Turán's theorem. The first lemma proves that upper bounds of the sizes of the partitions sum up exactly to the overall upper bound.

lemma *turan-sum-eq* :
fixes *n p* :: *nat*
assumes $p \geq 2$ **and** $p \leq n$
shows $(p-1) * (p-2) / 2 + (1 - 1 / (p-1)) * (n - p + 1)$ ^ $2 / 2 + (p - 2) * (n - p + 1) = (1 - 1 / (p-1)) * n$^$2 / 2$
using *assms* **by** (*simp add*: *field-simps eval-nat-numeral*)

The next fact proves that the upper bound of edges is monotonically increasing with the size of the biggest clique.

```
lemma turan-mono :
  fixes n p q :: nat
  assumes 0 < q and q < p and p ≤ n
  shows (1 − 1 / q) ∗ n^2 / 2 ≤ (1 − 1 / (p−1)) ∗ n^2 / 2
  using assms by (simp add: frac-le)
```

4 Turán's Graph Theorem

In this section we turn to the direct adaptation of Turán's original proof as presented by Aigner and Ziegler [1]

```
theorem turan :
  fixes p n :: nat
  assumes finite (uverts G)
    and uwellformed G and ∀ C p′. uclique C G p′ ⟶ p′ < p and p ≥ 2 and
card(uverts G) = n
  shows card (uedges G) ≤ (1 − 1 / (p−1)) ∗ n^2 / 2 using assms
proof (induction n arbitrary: G rule: less-induct)
  case (less n)
  then show ?case
  proof (cases n < p)
    case True
    show ?thesis
    proof (cases n)
      case 0
      with less True show ?thesis
        by (auto simp add: wellformed-uverts-0)
    next
      case (Suc n′)
      with True have (1 − 1 / real n) ≤ (1 − 1 / real (p − 1))
      by (metis diff-Suc-1 diff-left-mono inverse-of-nat-le less-Suc-eq-le linorder-not-less
list-decode.cases not-add-less1 plus-1-eq-Suc)
      moreover have real (card (uedges G)) ≤ (1 − 1 / real n) ∗ real (n²) / 2
        using ugraph-max-edges [OF less(3,6,2)]
      by (smt (verit, ccfv-SIG) left-diff-distrib mult.right-neutral mult-of-nat-commute
nonzero-mult-div-cancel-left of-nat-1 of-nat-mult power2-eq-square times-divide-eq-left)
      ultimately show ?thesis
      using Rings.ordered-semiring-class.mult-right-mono divide-less-eq-numeral1(1)
le-less-trans linorder-not-less of-nat-0-le-iff
        by (smt (verit, ccfv-threshold) divide-nonneg-nonneg times-divide-eq-right)
    qed
  next
    case False
    show ?thesis
    proof −
      obtain C q where C: uclique C G q
        and C-max: (∀ C q′. uclique C G q′ ⟶ q′ ≤ q)
        and q: q < card (uverts G)
```

using *clique-exists-gt0* [*OF* ‹*finite* (*uverts G*)›] *False* ‹$p \geq 2$› *less.prems*(*1*,*3*,*5*)
by (*metis card.empty card-gt-0-iff le-eq-less-or-eq order-less-le-trans pos2*)
obtain *E C′* **where** *E*: *uwellformed* (*uverts G*, *uedges G* ∪ *E*)
and *C′*: (*uclique C′* (*uverts G*, *uedges G* ∪ *E*) (*p*−*1*))
and *C′-max*: (∀ *C q′*. *uclique C* (*uverts G*, *uedges G* ∪ *E*) *q′* ⟶ *q′* ≤ *p*−*1*)
using *clique-add-edges-max* [*OF* ‹*finite* (*uverts G*)› ‹*uwellformed G*› *q* - *C-max*, *of p*−*1*]
using *C less*(*4*) *less*(*5*) *False* ‹*card* (*uverts G*) = *n*›
by (*smt* (*verit*) *One-nat-def Suc-leD Suc-pred less-Suc-eq-le linorder-not-less order-less-le-trans pos2*)
have *card* {*e* ∈ *uedges G* ∪ *E*. *e* ⊆ *uverts C′*} = (*p*−*1*) ∗ (*p*−*2*) / *2*
using *clique-edges-inside* [*OF E* - - - *C′*] *False less*(*2*) *less.prems*(*4*) *C′*
by (*smt* (*verit*, *del-insts*) *Collect-cong Suc-1 add-leD1 clique-max-size fst-conv of-nat-1 of-nat-add of-nat-diff of-nat-mult plus-1-eq-Suc snd-conv*)
moreover have *card* {*e* ∈ *uedges G* ∪ *E*. *e* ⊆ *uverts G* − *uverts C′*} ≤ (*1* − *1* / (*p*−*1*)) ∗ (*n* − *p* + *1*) ^ *2* / *2*
proof −
have *real*(*card*{*e* ∈ *uedges* (*uverts G*, *uedges G* ∪ *E*). *e* ⊆ *uverts* (*uverts G*, *uedges G* ∪ *E*) − *uverts C′*})
≤ (*1* − *1* / (*real p* − *1*)) ∗ (*real n* − *real p* + *1*)^2 / *2*
using *clique-edges-outside* [*OF E* - *less*(*5*) - - *C′ C′-max*, *of n*] *linorder-class.leI* [*OF False*] *less*(*1*,*2*,*6*)
by (*metis* (*no-types*, *lifting*) *fst-conv*)
then show *?thesis*
by (*simp*, *smt* (*verit*, *best*) *False One-nat-def Suc-1 Suc-leD add.commute leI less.prems*(*4*) *of-nat-1 of-nat-diff*)
qed
moreover have *card* {*e* ∈ *uedges G* ∪ *E*. *e* ∩ *uverts C′* ≠ {} ∧ *e* ∩ (*uverts G* − *uverts C′*) ≠ {}} ≤ (*p* − *2*) ∗ (*n* − *p* + *1*)
using *clique-edges-inside-to-outside* [*OF E* - - - - *C′ C′-max*, *of n*] *less*(*2*,*5*,*6*)
by (*simp*, *metis* (*no-types*, *lifting*) *C′ False Nat.add-diff-assoc Nat.add-diff-assoc2 One-nat-def Suc-1 clique-max-size fst-conv leI mult-Suc-right plus-1-eq-Suc*)
ultimately have *real* (*card* (*uedges G* ∪ *E*)) ≤ (*1* − *1* / *real* (*p* − *1*)) ∗ *real* (n^2) / *2*
using *graph-partition-edges-card* [*OF* - *E*, *of uverts C′*]
using *less*(*2*) *turan-sum-eq* [*OF* ‹*2* ≤ *p*›, *of n*] *False C′ uclique-def subgraph-def*
by (*smt* (*verit*) *Collect-cong fst-eqD linorder-not-le of-nat-add of-nat-mono snd-eqD*)
then show *?thesis*
using *less*(*2*) *E finite-verts-edges Finite-Set.card-mono* [*OF* - *Set.Un-upper1* [*of uedges G E*]]
by *force*
qed
qed
qed

5 A simplified proof of Turán's Graph Theorem

In this section we discuss a simplified proof of Turán's Graph Theorem which uses an idea put forward by the author: Instead of increasing the size of the biggest clique it is also possible to use the fact that the expression in Turán's graph theorem is monotonically increasing in the size of the biggest clique (Lemma *turan-mono*). Hence, it suffices to prove the upper bound for the actual biggest clique size in the graph. Afterwards, the monotonicity provides the desired inequality.

The simplifications in the proof are annotated accordingly.

theorem *turan′* :
  fixes *p n* :: *nat*
  assumes *finite* (*uverts G*)
    and *uwellformed G* **and** ∀ *C p′*. *uclique C G p′* ⟶ *p′* < *p* **and** *p* ≥ *2* **and** *card*(*uverts G*) = *n*
  shows *card* (*uedges G*) ≤ (*1* − *1* / (*p*−*1*)) ∗ *n*^*2* / *2* **using** *assms*
proof (*induction n arbitrary*: *p G rule*: *less-induct*)

In the simplified proof we also need to generalize over the biggest clique size p so that we can leverage the induction hypothesis in the proof for the already pre-existing biggest clique size which might be smaller than $p - 1$.

  case (*less n*)
  then show *?case*
  proof (*cases n* < *p*)
    case *True*
    show *?thesis*
    proof (*cases n*)
      case *0*
      with *less True* **show** *?thesis*
        by (*auto simp add*: *wellformed-uverts-0*)
    next
      case (*Suc n′*)
      with *True* **have** (*1* − *1* / *real n*) ≤ (*1* − *1* / *real* (*p* − *1*))
      by (*metis diff-Suc-1 diff-left-mono inverse-of-nat-le less-Suc-eq-le linorder-not-less list-decode.cases not-add-less1 plus-1-eq-Suc*)
      moreover have *real* (*card* (*uedges G*)) ≤ (*1* − *1* / *real n*) ∗ *real* (n^2) / *2*
        using *ugraph-max-edges* [*OF less*(*3*,*6*,*2*)]
      by (*smt* (*verit*, *ccfv-SIG*) *left-diff-distrib mult.right-neutral mult-of-nat-commute nonzero-mult-div-cancel-left of-nat-1 of-nat-mult power2-eq-square times-divide-eq-left*)
      ultimately show *?thesis*
      using *Rings.ordered-semiring-class.mult-right-mono divide-less-eq-numeral1*(*1*) *le-less-trans linorder-not-less of-nat-0-le-iff*
        by (*smt* (*verit*, *ccfv-threshold*) *divide-nonneg-nonneg times-divide-eq-right*)
    qed
  next
    case *False*
    show *?thesis*
    proof −

from *False* ‹$p \geq 2$›
obtain *C q* **where** *C*: *uclique C G q*
and *C-max*: $(\forall$ *C* q'. *uclique C G* $q' \longrightarrow q' \leq q)$
and *q1*: $q <$ *card* (*uverts G*) **and** *q2*: $0 < q$
and *pq*: $q < p$
using *clique-exists-gt0* [*OF* ‹*finite* (*uverts G*)›] *clique-exists1 less.prems*(*1*,*3*,*5*)
by (*metis card.empty card-gt-0-iff le-eq-less-or-eq order-less-le-trans pos2*)

In the unsimplified proof we extend this existing greatest clique C to a clique of size $p - 1$. This part is made superfluous in the simplified proof. In particular, also Section 2.2 is unneeded for this simplified proof. From here on the proof is analogous to the unsimplified proof with the potentially smaller clique of size q in place of the extended clique.

have *card* $\{e \in$ *uedges G*. $e \subseteq$ *uverts C*$\} = q * (q-1) / 2$
using *clique-edges-inside* [*OF less*(*3*,*2*) - - *C*] *q1 less*(*6*)
by *auto*
moreover have *card* $\{e \in$ *uedges G*. $e \subseteq$ *uverts G* $-$ *uverts C*$\} \leq (1 - 1 / q) * (n - q)$ ^ $2 / 2$
proof −
have *real* (*card* $\{e \in$ *uedges G*. $e \subseteq$ *uverts G* $-$ *uverts C*$\}$)
$\leq (1 - 1 / (real\ (q + 1) - 1)) * (real\ n - real\ (q + 1) + 1)^2 / 2$
using *clique-edges-outside* [*OF less*(*3*,*2*) - - , *of q+1 n C*] *C C-max q1 q2 linorder-class.leI* [*OF False*] *less*(*1*,*6*)
by (*smt* (*verit*, *ccfv-threshold*) *Suc-1 Suc-eq-plus1 Suc-leI diff-add-inverse2 zero-less-diff*)
then show *?thesis*
using *less.prems*(*5*) *q1*
by (*simp add*: *of-nat-diff*)
qed
moreover have *card* $\{e \in$ *uedges G*. $e \cap$ *uverts C* $\neq \{\} \wedge e \cap$ (*uverts G* $-$ *uverts C*) $\neq \{\}\} \leq (q - 1) * (n - q)$
using *clique-edges-inside-to-outside* [*OF less*(*3*,*2*) *q2* - *less*(*6*) *C C-max*] *q1*
by *simp*
ultimately have *real* (*card* (*uedges G*)) $\leq (1 - 1 /$ *real q*$) *$ *real* $(n^2) / 2$
using *graph-partition-edges-card* [*OF less*(*2*,*3*), *of uverts C*]
using *C uclique-def subgraph-def q1 q2 less.prems*(*5*) *turan-sum-eq* [*of Suc q n*]
by (*smt* (*verit*) *Nat.add-diff-assoc Suc-1 Suc-le-eq Suc-le-mono add.commute add.right-neutral diff-Suc-1 diff-Suc-Suc of-nat-add of-nat-mono plus-1-eq-Suc*)
then show *?thesis*

The final statement can then easily be derived with the monotonicity (Lemma *turan-mono*).

using *turan-mono* [*OF q2 pq*, *of n*] *False*
by *linarith*
qed
qed
qed

end