Maximum Cardinality Matching

Christine Rizkallah

September 19, 2015

Abstract

A matching in a graph $G$ is a subset $M$ of the edges of $G$ such that no two share an endpoint. A matching has maximum cardinality if its cardinality is at least as large as that of any other matching. An odd-set cover OSC of a graph $G$ is a labeling of the nodes of $G$ with integers such that every edge of $G$ is either incident to a node labeled 1 or connects two nodes labeled with the same number $i \geq 2$.

**Theorem 1** (Edmonds [2]). Let $M$ be a matching in a graph $G$ and let OSC be an odd-set cover of $G$. For any $i \geq 0$, let $n_i$ be the number of nodes labeled $i$. If

$$|M| = n_1 + \sum_{i \geq 2} \lfloor n_i/2 \rfloor$$

then $M$ is a maximum cardinality matching.

We provide an Isabelle proof of Edmonds theorem. For an explanation of the proof see [1].

**Contents**

1 Definitions 2

2 Lemmas 2

2.1 $|M_1| \leq n_1$ 3

2.2 $|M_i| \leq \lfloor n_i/2 \rfloor$ 6

2.3 $|M| \leq \sum |M_i|$ 6

3 Final Theorem 9

theory Matching

imports Main

begin

type-synonym label = nat

1
1 Definitions

definition finite-graph :: 'v set => ('v * 'v) set => bool where
finite-graph V E = (finite V ∧ finite E ∧
(∀ e ∈ E. fst e ∈ V ∧ snd e ∈ V ∧ fst e ~ = snd e))

definition degree :: ('v * 'v) set => 'v ⇒ nat where
degree E v = card { e ∈ E. fst e = v ∨ snd e = v }

definition edge-as-set :: ('v * 'v) => 'v set where
degree E = { fst e, snd e }

definition N :: 'v set => ('v ⇒ label) ⇒ nat ⇒ nat where
N V L i = card { v ∈ V. L v = i }

definition weight :: label set ⇒ (label ⇒ nat) ⇒ nat where
weight LV f = f 1 + (∑ i∈LV. (f i) div 2)

definition OSC :: ('v ⇒ label) ⇒ ('v * 'v) set ⇒ bool where
OSC L E = (∀ e ∈ E. L (fst e) = 1 ∨ L (snd e) = 1 ∨
L (fst e) = L (snd e) ∧ L (fst e) > 1)

definition disjoint-edges :: ('v * 'v) ⇒ ('v * 'v) ⇒ bool where
disjoint-edges e1 e2 = (fst e1 ≠ fst e2 ∧
snd e1 ≠ snd e2 ∧
snd e1 ≠ fst e2 ∧
snd e1 ≠ snd e2)

definition matching :: 'v set ⇒ ('v * 'v) set ⇒ ('v * 'v) set ⇒ bool where
matching V E M = (M ⊆ E ∧ finite-graph V E ∧
(∀ e1 ∈ M. ∀ e2 ∈ M. e1 ≠ e2 ⇒ disjoint-edges e1 e2))

definition matching-i :: nat ⇒ 'v set ⇒ ('v * 'v) set ⇒ ('v * 'v) set ⇒ ('v ⇒ label) ⇒ ('v * 'v) set where
matching-i i V E M L = { e ∈ M. i = 1 ∨ (L (fst e) = i ∨ L (snd e) = i) ∨
i > 1 ∧ L (fst e) = i ∧ L (snd e) = i }

definition V-i :: nat ⇒ 'v set ⇒ ('v * 'v) set ⇒ ('v * 'v) set ⇒ ('v ⇒ label) ⇒ 'v set where
V-i i V E M L = (edge-as-set V i V E M L)

definition endpoint-inV :: 'v set ⇒ ('v * 'v) ⇒ 'v where
endpoint-inV V e = (if fst e ∈ V then fst e else snd e)

definition relevant-endpoint :: ('v ⇒ label) ⇒ 'v set ⇒
('v * 'v) ⇒ 'v where
relevant-endpoint L V e = (if L (fst e) = 1 then fst e else snd e)

2 Lemmas

lemma definition-of-range:
endpoint\-in \ V1 \cdot \text{matching\-i} \ 1 \ V \ E \ M \ L = \{\ v. \ \exists\ e \in \text{matching\-i} \ 1 \ V \ E \ M \ L. \ \text{endpoint\-in} \ V1 \ e = v\} \ \text{by auto}

\textbf{lemma} matching\-i\-edges\-as\-sets:
\begin{align*}
\text{edge-as-set} \ \cdot \ \text{matching\-i} \ i \ V \ E \ M \ L = \\
\{\ e1. \ \exists\ (u, v) \in \text{matching\-i} \ i \ V \ E \ M \ L. \ \text{edge-as-set} (u, v) = e1\}\ \text{by auto}
\end{align*}

\textbf{lemma} matching\-disjointness:
\begin{align*}
\text{assumes matching} \ V \ E \ M \\
\text{assumes} e1 \in M \\
\text{assumes} e2 \in M \\
\text{assumes} e1 \neq e2 \\
\text{shows edge-as-set} e1 \cap \text{edge-as-set} e2 = \{\}
\end{align*}
\textbf{using} \text{assms}
\textbf{by} (auto \ \text{simp add: edge-as-set-def disjoint-edges-def matching-def})

\textbf{lemma} expand\-set\-containment:
\begin{align*}
\text{assumes matching} \ V \ E \ M \\
\text{assumes} e \in M \\
\text{shows} e \in E
\end{align*}
\textbf{using} \text{assms}
\textbf{by} (auto \ \text{simp add: matching-def})

\textbf{theorem} injectivity:
\begin{align*}
\text{assumes} \ \text{is-osc}: \ \text{OSC} \ L \ E \\
\text{assumes} \ \text{is-m}: \ \text{matching} \ V \ E \ M \\
\text{assumes} e1\text{-in-M1}: \ e1 \in \text{matching\-i} \ 1 \ V \ E \ M \ L \\
\text{and} e2\text{-in-M1}: \ e2 \in \text{matching\-i} \ 1 \ V \ E \ M \ L \\
\text{assumes} \ \text{diff}: \ (e1 \neq e2) \\
\text{shows} \ \text{endpoint\-in} \ \{v \in V. \ L v = 1\} \ e1 \neq \ \text{endpoint\-in} \ \{v \in V. \ L v = 1\} \ e2
\end{align*}
\textbf{proof} –
\begin{itemize}
\item from \text{e1\-in-M1} \ \text{have} \ e1 \in M \ \text{by} \ (auto \ \text{simp add: matching\-i-def})
\item moreover
\item from \text{e2\-in-M1} \ \text{have} \ e2 \in M \ \text{by} \ (auto \ \text{simp add: matching\-i-def})
\item ultimately
\item have disjoint-edges-sets: \text{edge-as-set} e1 \cap \text{edge-as-set} e2 = \{}
\item \textbf{using} \text{diff is-n matching-disjointness by fast}
\item then \textbf{show} \ ?\text{thesis by} (auto \ \text{simp add: edge-as-set-def endpoint\-in\-V-def})
\end{itemize}
\textbf{qed}

2.1 \ |M1| \leq n1

\textbf{lemma} card\-M1\-le\-NVL1:
\begin{align*}
\text{assumes matching} \ V \ E \ M \\
\text{assumes OSC} \ L \ E \\
\text{shows} \ \text{card} \ \text{matching\-i} \ 1 \ V \ E \ M \ L \ \leq \ ( NVL1)
\end{align*}
\textbf{proof} –
\begin{itemize}
\item let \ ?\text{if} = \ \text{endpoint\-in} \ \{v \in V. \ L v = 1\}
\end{itemize}
let \( A = \text{matching-i } i \ V \ E \ M \ L \)
let \( B = \{ v \in V. \ L v = 1 \} \)

have inj-on \( \text{if } A \text{ using assms injectivity} \)
  unfolding inj-on-def by blast

moreover have \( \text{if } i \ A \subseteq B \)

proof –
  
  \{ fix \( e \) assume \( e \in \text{matching-i } i \ V \ E \ M \ L \) 
  then have \( \text{endpoint-in } V \{ v \in V. \ L v = 1 \} \) e \( \in \{ v \in V. \ L v = 1 \} \) 
  using assms 
  by (auto simp add: endpoint-inV-def matching-def matching-i-def OSC-def finite-graph-def definition-of-range) 
  
  \}

  then show \( \text{thesis using assms definition-of-range by blast} \)

qed

moreover have \( \text{finite } B \text{ using assms} \)
  by (simp add: matching-def finite-graph-def)

ultimately show \( \text{thesis unfolding } N \text{-def by (rule card-inj-on-le)} \)

qed

lemma edge-as-set-inj-on-Mi:
  assumes \( \text{matching } V \ E \ M \)
  shows \( \text{inj-on edge-as-set } (\text{matching-i } i \ V \ E \ M \ L) \)
  using assms
  unfolding inj-on-def edge-as-set-def matching-def
  disjoint-edges-def matching-i-def
  by blast

lemma card-Mi-eq-card-edge-as-set-Mi:
  assumes \( \text{matching } V \ E \ M \)
  shows \( \text{card } (\text{matching-i } i \ V \ E \ M \ L) = \text{card } (\text{edge-as-set'}\ \text{matching-i } i \ V \ E \ M \ L) \)
  (is \( \text{card } ?Mi = \text{card } (?f ' -)\))

proof –
  from assms have bij-betw \( \text{if } \ ?Mi \) \( (?f ' \ ?Mi) \)
  by (simp add: bij-betw-def matching-i-edges-as-sets edge-as-set-inj-on-Mi)
  then show \( \text{thesis by (rule bij-betw-same-card)} \)

qed

lemma card-edge-as-set-Mi-twice-card-partitions:
  assumes \( \text{OSC } L \ E \land \text{matching } V \ E \ M \land i > 1 \)
  shows \( 2 \ast \text{card } (\text{edge-as-set'}\ \text{matching-i } i \ V \ E \ M \ L) \)
  \( = \text{card } (V-i \ i \ V \ E \ M \ L) \) (is \( 2 \ast \text{card } ?C = \text{card } ?Vi) \)

proof –
  from assms have \( 1: \text{finite } (\bigcup ?C) \)
  by (auto simp add: matching-def finite-graph-def
  matching-i-def edge-as-set-def finite-subset)
  show \( \text{thesis unfolding } V-i \text{-def} \)
  proof (rule card-partition)
    show \( \text{finite } ?C \text{ using } 1 \text{ by (rule finite-UnionD)} \)

  qed
next
  show finite $(\bigcup \mathcal{C})$ using 1.
next
fix \(c\) assume \(c \in \mathcal{C}\) then show \(\text{card } c = 2\)
proof (rule imageE)
  fix \(x\)
  assume 2: \(c = \text{edge-as-set } x\) and 3: \(x \in \text{matching-i } i \ V \ E \ M \ L\)
  with assms have \(x \in E\)
  unfolding matching-i-def matching-def by blast
  then have \(\text{fst } x \neq \text{snd } x\) using assms 3
  by (auto simp add: matching-def finite-graph-def)
  with 2 show \(?thesis\) by (auto simp add: edge-as-set-def)
qed
next
fix \(x_1 \ x_2\)
assume 4: \(x_1 \in \mathcal{C}\) and 5: \(x_2 \in \mathcal{C}\) and 6: \(x_1 \neq x_2\)
{ 
  fix \(e_1 \ e_2\)
  assume 7: \(x_1 = \text{edge-as-set } e_1 \ e_1 \in \text{matching-i } i \ V \ E \ M \ L\)
  \(x_2 = \text{edge-as-set } e_2 \ e_2 \in \text{matching-i } i \ V \ E \ M \ L\)
  from assms have \(\text{matching } V \ E \ M\) by simp
  moreover
  from 7 assms have \(e_1 \in M\) and \(e_2 \in M\)
  by (simp-all add: matching-i-def)
  moreover from 6 7 have \(e_1 \neq e_2\) by blast
  ultimately have \(x_1 \cap x_2 = \{\}\) unfolding 7
  by (rule matching-disjointness)
} 
with 4 5 show \(x_1 \cap x_2 = \{\}\) by clarsimp
qed
lemma card-Mi-twice-card-Vi:
assumes \(\text{OSC } L \ E \land \text{matching } V \ E \ M \land i > 1\)
shows \(2 \ast \text{card } (\text{matching-i } i \ V \ E \ M \ L) = \text{card } (V-i i \ V \ E \ M \ L)\)
proof
from assms have \(\text{finite } (V-i i \ V \ E \ M \ L)\)
by (auto simp add: edge-as-set-def finite-subset
  matching-def finite-graph-def V-i-def matching-i-def)
with assms show \(?thesis\)
by (simp add: card-Mi-eq-card-edge-as-set-Mi
  card-edge-as-set-Mi-twice-card-partitions V-i-def)
qed
lemma card-Mi-le-floor-div-2-Vi:
assumes \(\text{OSC } L \ E \land \text{matching } V \ E \ M \land i > 1\)
shows \(\text{card } (\text{matching-i } i \ V \ E \ M \ L) \leq (\text{card } (V-i i \ V \ E \ M \ L)) \div 2\)
using card-Mi-twice-card-Vi[OF assms]
by arith
lemma card-Vi-le-NVLi:
  assumes i > 1 ∧ matching V E M
  shows card (V-i i V E M L) ≤ N V L i
unfolding N-def
proof (rule card-mono)
  show finite {v ∈ V. L v = i} using assms
    by (simp add: matching-def finite-graph-def)
next
  let ?A = edge-as-set ' matching-i i V E M L
  let ?C = {v ∈ V. L v = i}
  show V-i i V E M L ⊆ ?C using assms unfolding V-i-def
proof (intro Union-least)
  fix X assume X ∈ ?A
  with assms have ∃ x ∈ matching-i i V E M L. edge-as-set x = X
    by (simp add: matching-i-edges-as-sets)
  with assms show X ⊆ ?C
    unfolding finite-graph-def matching-def
    matching-i-def edge-as-set-def by blast
qed

2.2 |M_i| ≤ \lfloor n_i/2 \rfloor

lemma card-Mi-le-floor-div-2-NVLi:
  assumes OSC L E ∧ matching V E M ∧ i > 1
  shows card (matching-i i V E M L) ≤ (N V L i) div 2
proof
  let ?UnMi = \bigcup x ∈ L'V. matching-i x V E M L
  from assms have card (V-i i V E M L) ≤ (N V L i)
    by (simp add: card-Vi-le-NVLi)
  then have card (V-i i V E M L) div 2 ≤ (N V L i) div 2
    by simp
  moreover from assms have
card (matching-i i V E M L) ≤ card (V-i i V E M L) div 2
    by (intro card-Mi-le-floor-div-2-Vi)
  ultimately show ?thesis by auto
qed

2.3 |M| ≤ \sum |M_i|

lemma card-M-le-sum-card-Mi:
  assumes matching V E M and OSC L E
  shows card M ≤ (\sum i ∈ L'V. card (matching-i i V E M L))
    (is card - ≤ ?CardMi)
proof
  let ?UnMi = \bigcup x ∈ L'V. matching-i x V E M L
  from assms have 1: finite ?UnMi
    by (auto simp add: matching-def
      finite-graph-def matching-i-def finite-subset)
  {
fix e assume e-inM: e ∈ M
let ?v = relevant-endpoint L V e
have 1: e ∈ matching-i (L ?v) V E M L using assms e-inM
proof cases
  assume L (fst e) = 1
  thus ?thesis using assms e-inM
  by (simp add: relevant-endpoint-def matching-i-def)
next
  assume a: L (fst e) ≠ 1
  have L (fst e) = 1 ∨ L (snd e) = 1
    ∨ (L (fst e) = L (snd e) ∧ L (fst e) > 1)
    using assms e-inM unfolding OSC-def
    by (blast intro: expand-set-containment)
  thus ?thesis using assms e-inM a
  by (auto simp add: relevant-endpoint-def matching-i-def)
qed
have 2: ?v ∈ V using assms e-inM
  by (auto simp add: matching-def relevant-endpoint-def finite-graph-def)
then have ∃ v ∈ V. e ∈ matching-i (L v) V E M L using assms 1 2
  by (intro bexI)
}
with assms have M ⊆ ?UnMi by (auto)
with assms and 1 have card M ≤ card ?UnMi by (intro card-mono)
moreover from assms have card ?UnMi = ?CardMi
proof (intro card-UN-disjoint)
  show finite (L′V) using assms
  by (simp add: matching-def finite-graph-def)
next
  show ∀ i∈L′V. finite (matching-i i V E M L) using assms
    unfolding matching-def finite-graph-def matching-i-def
    by (blast intro: finite-subset)
next
  show ∀ i ∈ L′V. ∀ j ∈ L′V. i ≠ j −→
    matching-i i V E M L ∩ matching-i j V E M L = {}
    using assms
  by (auto simp add: matching-i-def)
qed
ultimately show ?thesis by simp
qed

theorem card-M-le-weight-NVLi:
  assumes matching V E M and OSC L E
  shows card M ≤ weight {i ∈ L′ V. i > 1} (N V L) (is - ≤ ?W)
proof —
  let ?M01 = ∑ i | i ∈ L′V ∧ (i=1 ∨ i=0). card (matching-i i V E M L)
  let ?Mgr1 = ∑ i | i ∈ L′V ∧ 1 < i. card (matching-i i V E M L)
  let ?Mi = ∑ i L′V. card (matching-i i V E M L)
  have card M ≤ ?Mi using assms by (rule card-M-le-sum-card-Mi)
moreover
have \( \forall i \in L \setminus V. i = 1 \lor i = 0 \)

proof

let \( A = \{ i \in L \setminus V. i = 1 \lor i = 0 \} \)

let \( B = \{ i \in L \setminus V. i < 1 \} \)

let \( g = \lambda i. \text{card} (\text{matching-i} i V E M L) \)

let \( set01 = \{ i, i : L \setminus V \& (i = 1 \lor i = 0) \} \)

have \( a : L \setminus V = A \cup B \) using assms by auto

have \( \text{finite} V \) using assms by (simp add: matching-def finite-graph-def)

have \( b : \sum g (A \cup B) = \sum g A + \sum g B \) using assms (finite V) by (auto intro: setsum.union-disjoint)

have \( 1 : \forall i = ?M01 + ?Mgr1 \) using assms a b

by (simp add: matching-def finite-graph-def)

moreover

have \( 0 : \text{card} (\text{matching-i} 0 V E M L) = 0 \) using assms

by (simp add: matching-i-def)

have \( 2 : ?M01 \leq NVL1 \)

proof cases

assume \( a : 1 \in L \setminus V \)

have \( ?M01 = \text{card} (\text{matching-i} 1 V E M L) \)

proof cases

assume \( b : 0 \in L \setminus V \)

with \( a \) have \( set01 = \{ 0, 1 \} \) by blast

thus \( \text{thesis} \) using assms 0 by simp

next

assume \( b : 0 \notin L \setminus V \)

with \( a \) have \( set01 = \{ 1 \} \) by (auto simp del:One-nat-def)

thus \( \text{thesis} \) by simp

qed

thus \( \text{thesis} \) using assms a

by (simp del:One-nat-def, intro card-M1-le-NVL1)

next

assume \( a : 1 \notin L \setminus V \)

show \( \text{thesis} \)

proof cases

assume \( b : 0 \in L \setminus V \)

with \( a \) have \( set01 = \{ 0 \} \) by (auto simp del:One-nat-def)

thus \( \text{thesis} \) using assms 0 by auto

next

assume \( b : 0 \notin L \setminus V \)

with \( a \) have \( set01 = \{ \} \) by (auto simp del:One-nat-def)

then have \( ?M01 = (\sum i \in \{ \}. \text{card} (\text{matching-i} i V E M L)) \) by auto

thus \( \text{thesis} \) by simp

qed

qed

moreover

have \( 3 : ?Mgr1 \leq (\sum i \in L \setminus V \& 1 < i. NVL1 i \div 2) \) using assms

by (intro setsum-mono card-Mi-le-floor-div-2-NVL1, simp)

ultimately
show \textit{thesis} using 1 2 3 assms by (simp add: weight-def)
qed
ultimately show \textit{thesis} by simp
qed

3 Final Theorem

The following theorem is due to Edmond [2]:

\textbf{theorem} maximum-cardinality-matching:
assumes matching $V$ $E$ $M$ and OSC $L$ $E$
and card $M$ = weight \{i \in L \mid V. i > 1\} (N $V$ $L$)
and matching $V$ $E$ $M'$
shows card $M'$ $\leq$ card $M$
using assms card-M-le-weight-NVLi
by simp

The widely used algorithmic library LEDA has a certifying algorithm for maximum cardinality matching. This Isabelle proof is part of the work done to verify the checker of this certifying algorithm. For more information see [1].

end

References
