# Verified Approximation Algorithms 

Robin Eßmann, Tobias Nipkow, Simon Robillard, Ujkan Sulejmani

September 13, 2023


#### Abstract

We present the first formal verifications of approximation algorithms for NP-complete optimization problems: vertex cover, set cover, independent set, center selection, load balancing, and bin packing. The proofs correct incompletnesses in existing proofs and improve the approximation ratio in one case. A detailed description of our work (excluding center selection) has been published in the proceedings of IJCAR 2020 [3].


## Contents

1 Vertex Cover ..... 2
1.1 Graph ..... 2
1.2 The Approximation Algorithm ..... 2
1.3 Version for Hypergraphs ..... 4
2 Set Cover ..... 6
3 Independent Set ..... 14
3.1 Graph ..... 14
3.2 Wei's algorithm: $(\Delta+1)$-approximation ..... 16
3.3 Wei's algorithm: $\Delta$-approximation ..... 19
3.4 Wei's algorithm with dynamically computed approximation ratio ..... 24
4 Load Balancing ..... 26
4.1 Formalization of a Correct Load Balancing ..... 28
4.1.1 Definition ..... 28
4.1.2 Lower Bounds for the Makespan ..... 29
4.2 The Greedy Approximation Algorithm ..... 30
5 Bin Packing ..... 39
5.1 Formalization of a Correct Bin Packing ..... 39
5.2 The Proposed Approximation Algorithm ..... 40
5.2.1 Functional Correctness ..... 40
5.2.2 Lower Bounds for the Bin Packing Problem ..... 47
5.2.3 Proving the Approximation Factor ..... 53
5.3 The Full Linear Time Version of the Proposed Algorithm ..... 60
6 Center Selection ..... 78
6.1 A Preliminary Algorithm and Proof ..... 81
6.2 The Main Algorithm ..... 85

## 1 Vertex Cover

```
theory Approx-VC-Hoare
imports HOL-Hoare.Hoare-Logic
begin
```

The algorithm is classical, the proof is based on and augments the one by Berghammer and Müller-Olm [1].

### 1.1 Graph

A graph is simply a set of edges, where an edge is a 2 -element set.

```
definition vertex-cover :: 'a set set }=>\mathrm{ 'a set }=>\mathrm{ bool where
vertex-cover E C=(\foralle\inE.e\capC\not={})
```

abbreviation matching :: 'a set set $\Rightarrow$ bool where
matching $M \equiv$ pairwise disjnt $M$
lemma card-matching-vertex-cover:
$\llbracket$ finite $C$; matching $M ; M \subseteq E ;$ vertex-cover $E C \rrbracket \Longrightarrow \operatorname{card} M \leq \operatorname{card} C$ apply (erule card-le-if-inj-on-rel[where $r=\lambda e v . v \in e])$
apply (meson disjnt-def disjnt-iff vertex-cover-def subsetCE)
by (meson disjnt-iff pairwise-def)

### 1.2 The Approximation Algorithm

Formulated using a simple(!) predefined Hoare-logic. This leads to a streamlined proof based on standard invariant reasoning.

The nondeterministic selection of an element from a set $F$ is simulated by SOME $x . x \in F$. The SOME operator is built into HOL: SOME $x . P x$ denotes some $x$ that satisfies $P$ if such an $x$ exists; otherwise it denotes an arbitrary element. Note that there is no actual nondeterminism involved: SOME $x . P x$ is some fixed element but in general we don't know which one. Proofs about SOME are notoriously tedious. Typically it involves showing first that $\exists x$. $P x$. Then $\exists x$. ? $P x \Longrightarrow$ ? $P(S O M E x$. ? $P x$ ) implies $P$ (SOME x. $P x$ ). There are a number of (more) useful related theorems: just click on $\exists x$. ? $P x \Longrightarrow$ ? $P(S O M E x$. ? $P x)$ to be taken there.

Convenient notation for choosing an arbitrary element from a set:

```
abbreviation some A \equivSOME x. x \in A
```

```
locale Edges =
    fixes \(E\) :: 'a set set
    assumes finE: finite \(E\)
    assumes edges2: \(e \in E \Longrightarrow\) card \(e=2\)
begin
```

The invariant:
definition inv-matching C F $M=$
(matching $M \wedge M \subseteq E \wedge$ card $C \leq 2 *$ card $M \wedge(\forall e \in M . \forall f \in F . e \cap f=$ \{\}))
definition invar :: 'a set $\Rightarrow$ 'a set set $\Rightarrow$ bool where
invar $C F=(F \subseteq E \wedge$ vertex-cover $(E-F) C \wedge$ finite $C \wedge(\exists M$. inv-matching $C$ F M) )

Preservation of the invariant by the loop body:

```
lemma invar-step:
    assumes F\not={} invar C F
    shows invar (C\cup some F)(F-{\mp@subsup{e}{}{\prime}\inF. some F\cap ' ( 
proof -
    from assms(2) obtain M where F\subseteqE and vc: vertex-cover (E-F)C and
fC: finite C
    and m: matching M M\subseteqE and card: card C\leq2 * card M
    and disj: }\foralle\inM.\forallf\inF.e\capf={
    by (auto simp: invar-def inv-matching-def)
    let ?e = SOME e. e\inF
    have ?e }\inF\mathrm{ using <F # {}> by (simp add: some-in-eq)
    hence fe': finite ?e using <F\subseteqE\rangle edges2 by(intro card-ge-0-finite) auto
    have ?e }\not\inM\mathrm{ using edges2〈?e }\inF\rangle\mathrm{ disj }\langleF\subseteqE\rangle\mathrm{ by fastforce
    have card': card ( }C\cup?,e)\leq2* card (insert ?e M
        using〈?e }\inF\rangle\langle?e \not\existsM\rangle card-Un-le[of C ?e]\langleF\subseteqE\rangle edges2 card fi
nite-subset[OF m(2) finE]
    by fastforce
    let ?M=M\cup{?e}
    have vc': vertex-cover ( }E-(F-{\mp@subsup{e}{}{\prime}\inF.?e\cap\mp@subsup{e}{}{\prime}\not={}}))(C\cup?e
        using vc by(auto simp: vertex-cover-def)
    have m': inv-matching ( }C\cup?e)(F-{\mp@subsup{e}{}{\prime}\inF\mathrm{ . ?e }\cap\mp@subsup{e}{}{\prime}\not={}})?
        using m card'}\langleF\subseteqE\rangle\langle?e\inF\rangledis
        by(auto simp: inv-matching-def Int-commute disjnt-def pairwise-insert)
    show ?thesis using }\langleF\subseteqE\ranglev\mp@subsup{c}{}{\prime}fCf\mp@subsup{e}{}{\prime}\mp@subsup{m}{}{\prime}\mathbf{by}(\mathrm{ auto simp add: invar-def Let-def)
qed
```

lemma approx-vertex-cover:
VARS C F
\{True\}

```
    C:={};
    F:=E;
    WHILE F}\not={
    INV {invar C F}
    DO C :=C U some F;
        F:=F-{\mp@subsup{e}{}{\prime}\inF. some F\cap 的}\not={}
    OD
    {vertex-cover E C ^(\forallC'. finite C'^ vertex-cover E C'}\longrightarrow\mathrm{ card C \2 * card
C')}
proof (vcg, goal-cases)
    case (1 C F)
    have inv-matching {} E {} by (auto simp add: inv-matching-def)
    with 1 show ?case by (auto simp add: invar-def vertex-cover-def)
next
    case (2 C F)
    thus ?case using invar-step[of F C] by(auto simp: Let-def)
next
    case (3 C F)
    then obtain M :: 'a set set where
        post: vertex-cover E C matching M M\subseteqE card C \leq2 * card M
        by(auto simp: invar-def inv-matching-def)
    have opt: card C\leq2 * card C' if C': finite C' vertex-cover E C' for C'
    proof -
        note post(4)
        also have 2 * card M\leq2 * card C'
        using card-matching-vertex-cover[OF C'(1) post(2,3) C'(2)] by simp
        finally show card C\leq2 * card C'.
    qed
    show ?case using post(1) opt by auto
qed
end
```


### 1.3 Version for Hypergraphs

Almost the same. We assume that the degree of every edge is bounded.

```
locale Bounded-Hypergraph \(=\)
    fixes \(E::\) 'a set set
    fixes \(k::\) nat
    assumes finE: finite \(E\)
    assumes edge-bnd: \(e \in E \Longrightarrow\) finite \(e \wedge\) card \(e \leq k\)
    assumes \(E 1:\{ \} \notin E\)
begin
definition inv-matching \(C F M=\)
    (matching \(M \wedge M \subseteq E \wedge \operatorname{card} C \leq k * \operatorname{card} M \wedge(\forall e \in M . \forall f \in F . e \cap f=\)
\{\}))
```

definition invar :: 'a set $\Rightarrow$ 'a set set $\Rightarrow$ bool where
invar $C F=(F \subseteq E \wedge$ vertex-cover $(E-F) C \wedge$ finite $C \wedge(\exists M$. inv-matching $C$ F M) )

```
lemma invar-step:
    assumes \(F \neq\{ \}\) invar \(C F\)
    shows invar \((C \cup\) some \(F)\left(F-\left\{e^{\prime} \in F\right.\right.\). some \(\left.\left.F \cap e^{\prime} \neq\{ \}\right\}\right)\)
proof -
    from \(\operatorname{assms}(2)\) obtain \(M\) where \(F \subseteq E\) and \(v c\) : vertex-cover \((E-F) C\) and
\(f C\) : finite \(C\)
            and m: matching \(M M \subseteq E\) and card: card \(C \leq k *\) card \(M\)
            and disj: \(\forall e \in M . \forall f \in F . e \cap f=\{ \}\)
    by (auto simp: invar-def inv-matching-def)
    let \(? e=S O M E\) e. \(e \in F\)
    have \(? e \in F\) using \(\langle F \neq\{ \}\rangle\) by (simp add: some-in-eq)
    hence \(f e^{\prime}\) : finite ?e using \(\langle F \subseteq E\rangle\) assms(2) edge-bnd by blast
    have \(? e \notin M\) using \(E 1\langle ? e \in F\rangle\) disj \(\langle F \subseteq E\rangle\) by fastforce
    have \(\operatorname{card}^{\prime}: \operatorname{card}(C \cup ? e) \leq k * \operatorname{card}\) (insert ? e M)
            using \(\langle ? e \in F\rangle\langle ? e \notin M\rangle\) card-Un-le[of \(C ? e]\langle F \subseteq E\rangle\) edge-bnd card fi-
nite-subset[OF m(2) finE]
    by fastforce
    let \(? M=M \cup\{?, e\}\)
    have \(v c^{\prime}:\) vertex-cover \(\left(E-\left(F-\left\{e^{\prime} \in F\right.\right.\right.\). \(\left.\left.\left.e, \cap e^{\prime} \neq\{ \}\right\}\right)\right)(C \cup ? e)\)
    using vc by (auto simp: vertex-cover-def)
    have \(m^{\prime}\) : inv-matching \((C \cup ? e)\left(F-\left\{e^{\prime} \in F\right.\right.\). ? \(\left.\left.e \cap e^{\prime} \neq\{ \}\right\}\right) ? M\)
    using \(m\) card \(^{\prime}\langle F \subseteq E\rangle\langle ? e \in F\rangle\) disj
    by (auto simp: inv-matching-def Int-commute disjnt-def pairwise-insert)
    show ?thesis using \(\langle F \subseteq E\rangle v c^{\prime} f C f e^{\prime} m^{\prime}\) by (auto simp add: invar-def Let-def)
qed
```

lemma approx-vertex-cover-bnd:
VARS C F
\{True $\}$
$C:=\{ \} ;$
$F:=E$;
WHILE $F \neq\{ \}$
INV \{invar $C F\}$
DO $C:=C \cup$ some $F$;
$F:=F-\left\{e^{\prime} \in F\right.$. some $\left.F \cap e^{\prime} \neq\{ \}\right\}$
OD
$\left\{\right.$ vertex-cover $E C \wedge\left(\forall C^{\prime}\right.$. finite $C^{\prime} \wedge$ vertex-cover $E C^{\prime} \longrightarrow$ card $C \leq k *$ card
$C^{\prime}$ ) $\}$
proof (vcg, goal-cases)
case (1 C F)
have inv-matching $\} E\}$ by (auto simp add: inv-matching-def)
with 1 show ?case by (auto simp add: invar-def vertex-cover-def)
next

```
    case (2 C F)
    thus ?case using invar-step[of F C] by(auto simp: Let-def)
next
    case (3 C F)
    then obtain M :: 'a set set where
        post: vertex-cover E C matching MM\subseteqE card C\leqk* card M
        by(auto simp: invar-def inv-matching-def)
    have opt: card C\leqk* card C''if C': finite C' vertex-cover E C' for }\mp@subsup{C}{}{\prime
    proof -
    note post(4)
    also have k* card M\leqk* card C'
    using card-matching-vertex-cover[OF C'(1) post(2,3) C'(2)] by simp
    finally show card C\leqk* card C'.
    qed
    show ?case using post(1) opt by auto
qed
end
end
```


## 2 Set Cover

theory Approx-SC-Hoare
imports
HOL-Hoare.Hoare-Logic
Complex-Main

## begin

This is a formalization of the set cover algorithm and proof in the book by Kleinberg and Tardos [4].
definition harm :: nat $\Rightarrow{ }^{\prime} a$ :: real-normed-field where
harm $n=\left(\sum k=1 . . n\right.$. inverse (of-nat $\left.\left.k\right)\right)$
locale Set-Cover $=$
fixes $w:$ nat $\Rightarrow$ real and $m::$ nat and $S::$ nat $\Rightarrow{ }^{\prime}$ a set
assumes $S$-finite: $\forall i \in\{1$.. $m\}$. finite $(S i)$
and $w$-nonneg: $\forall i .0 \leq w i$
begin
definition $U$ :: 'a set where

$$
U=(\bigcup i \in\{1 . . m\} . S i)
$$

lemma $S$-subset: $\forall i \in\{1 . . m\} . S i \subseteq U$
using $U$-def by blast
lemma $U$-finite: finite $U$
unfolding $U$-def using $S$-finite by blast
lemma empty-cover: $m=0 \Longrightarrow U=\{ \}$
using $U$-def by simp
definition sc :: nat set $\Rightarrow$ 'a set $\Rightarrow$ bool where
$s c C X \longleftrightarrow C \subseteq\{1 . . m\} \wedge(\bigcup i \in C . S i)=X$
definition cost $::$ 'a set $\Rightarrow$ nat $\Rightarrow$ real where
$\operatorname{cost} R i=w i / \operatorname{card}(S i \cap R)$
lemma cost-nonneg: $0 \leq$ cost $R i$ using w-nonneg by (simp add: cost-def)
cost $R i=0$ if card $(S i \cap R)=0$ ! Needs to be accounted for separately in min-arg.

```
fun min-arg \(::\) 'a set \(\Rightarrow\) nat \(\Rightarrow\) nat where
    \(\min -\arg R 0=1\)
\(\mid \min -\arg R(\) Suc \(x)=\)
    (let \(j=\) min-arg \(R x\)
    in if \(S j \cap R=\{ \} \vee(S(\) Suc \(x) \cap R \neq\{ \} \wedge \operatorname{cost} R(\) Suc \(x)<\operatorname{cost} R j)\) then
```

(Suc $x$ ) else $j$ )
lemma min-in-range: $k>0 \Longrightarrow$ min-arg $R k \in\{1 . . k\}$
by (induction $k$ ) (force simp: Let-def) +
lemma min-empty: $S($ min-arg $R k) \cap R=\{ \} \Longrightarrow \forall i \in\{1 . . k\} . S i \cap R=\{ \}$
proof (induction $k$ )
case (Suc k)
from Suc.prems have prem: $S$ (min-arg $R k) \cap R=\{ \}$ by (auto simp: Let-def
split: if-splits)
with $S u c . I H$ have $I H: \forall i \in\{1 . . k\} . S i \cap R=\{ \}$.
show ?case proof fix $i$ assume $i \in\{1$..Suc $k\}$ show $S i \cap R=\{ \}$
proof (cases $\langle i=S u c k\rangle$ )
case True with Suc.prems prem show?thesis by simp
next
case False with $I H\langle i \in\{1$..Suc $k\}\rangle$ show ?thesis by simp
qed
qed
qed $\operatorname{simp}$
lemma min-correct: $\llbracket i \in\{1 \ldots k\} ; S i \cap R \neq\{ \} \rrbracket \Longrightarrow \operatorname{cost} R($ min-arg $R k) \leq$ cost
Ri
proof (induction $k$ )
case (Suc k)
show ?case proof (cases $\langle i=S u c k\rangle)$

```
        case True with Suc.prems show ?thesis by (auto simp: Let-def)
    next
    case False with Suc.prems Suc.IH have IH: cost R (min-arg R k)\leqcost R i
by simp
    from Suc.prems False min-empty[of R k] have S(min-arg R k) \capR\not={} by
force
    with IH show ?thesis by (auto simp: Let-def)
    qed
qed simp
```

Correctness holds quite trivially for both $\mathrm{m}=0$ and $\mathrm{m}>0$ (assuming a set cover can be found at all, otherwise algorithm would not terminate).

```
lemma set-cover-correct:
VARS ( \(R\) :: 'a set) ( \(C\) :: nat set) ( \(i\) :: nat)
    \{True\}
    \(R:=U ; C:=\{ \}\)
    WHILE \(R \neq\{ \}\) INV \(\{R \subseteq U \wedge\) sc \(C(U-R)\} D O\)
    \(i:=\min -\arg R \mathrm{~m}\);
    \(R:=R-S i ;\)
    \(C:=C \cup\{i\}\)
    \(O D\)
    \(\{s c C U\}\)
proof (vcg, goal-cases)
    case 2 show ?case proof (cases m)
    case 0
    from empty-cover[OF this] 2 show ?thesis by (auto simp: sc-def)
    next
        case \(S u c\) then have \(m>0\) by simp
            from min-in-range \([O F\) this \(] 2\) show ?thesis using \(S\)-subset by (auto simp:
sc-def)
    qed
qed (auto simp: sc-def)
definition \(c\)-exists \(::\) nat set \(\Rightarrow\) ' \(a\) set \(\Rightarrow\) bool where
    \(c\)-exists \(C R=(\exists c\). sum \(w C=\operatorname{sum} c(U-R) \wedge(\forall i .0 \leq c i)\)
            \(\wedge(\forall k \in\{1 . . m\}\). sum \(c(S k \cap(U-R))\)
                        \(\leq\left(\sum j=\operatorname{card}(S k \cap R)+1 . . \operatorname{card}(S k)\right.\). inverse \(\left.\left.\left.j\right) * w k\right)\right)\)
definition inv :: nat set \(\Rightarrow\) 'a set \(\Rightarrow\) bool where
    inv \(C R \longleftrightarrow s c C(U-R) \wedge R \subseteq U \wedge c\)-exists \(C R\)
lemma invI:
    assumes sc \(C(U-R) R \subseteq U\)
            \(\exists c . \operatorname{sum} w C=\operatorname{sum} c(U-R) \wedge(\forall i .0 \leq c i)\)
        \(\wedge(\forall k \in\{1 . . m\}\). sum \(c(S k \cap(U-R))\)
                            \(\leq\left(\sum j=\operatorname{card}(S k \cap R)+1 . . c a r d(S k)\right.\). inverse \(\left.\left.j\right) * w k\right)\)
    shows inv \(C R\) using assms by (auto simp: inv-def c-exists-def)
lemma \(i n v D\) :
```

```
    assumes inv C R
    shows sc C (U-R)R\subseteqU
        \existsc. sum wC = sum c(U-R)^(\foralli.0\leqci)
    \wedge(\forallk\in{1..m}. sum c (Sk\cap(U-R))
    \leq(\sumj=\operatorname{card}(Sk\capR)+1..card (Sk). inverse j)*wk)
    using assms by (auto simp: inv-def c-exists-def)
lemma inv-init: inv {} U
proof (rule invI, goal-cases)
    case 3
    let ?c = (\lambda-. 0) :: 'a m real
    have sum w{} = sum?c(U-U) by simp
    moreover {
    have }\forallk\in{1..m}.0\leq(\sumj=\operatorname{card}(Sk\capU)+1..card (Sk). inverse j)
w}
            by (simp add: sum-nonneg w-nonneg)
    then have ( }\forallk\in{1..m}. sum ?c (Sk\cap(U-U)
                    \leq(\sumj=\operatorname{card}(Sk\capU)+1..card (Sk). inverse j)*wk) by simp
}
    ultimately show? ?case by blast
qed (simp-all add: sc-def)
lemma inv-step:
    assumes inv C R R\not={}
    defines [simp]: i\equivmin-arg R m
    shows inv (C\cup{i})(R-(S i))
proof (cases m)
    case 0
    from empty-cover[OF this] invD(2)[OF assms(1)] have R={} by blast
    then show ?thesis using assms(2) by simp
next
    case Suc then have 0<m by simp
    note hyp = invD[OF assms(1)]
    show ?thesis proof (rule invI, goal-cases)
        - Correctness
    case 1 have i\in{1..m} using min-in-range[OF <0<m>] by simp
    with hyp(1) S-subset show ?case by (auto simp: sc-def)
    next
        case 2 from hyp(2) show ?case by auto
    next
    case 3
        - Set Cover grows
    have }\existsi\in{1..m}.Si\capR\not={
        using assms(2) U-def hyp(2) by blast
    then have Si}\capR\not={} using min-empty by aut
    then have 0< card (Si\capR)
        using S-finite min-in-range[OF < 0 < m`] by auto
        - Proving properties of cost function
```

from hyp(3) obtain $c$ where sum $w C=\operatorname{sum} c(U-R) \forall i .0 \leq c i$ and $S U M: \forall k \in\{1 . . m\}$. sum $c(S k \cap(U-R))$
$\leq\left(\sum j=\operatorname{card}(S k \cap R)+1 . . \operatorname{card}(S k)\right.$. inverse $\left.j\right) * w k$ by blast
let $? c=(\lambda x$. if $x \in S i \cap R$ then cost $R$ i else $c x)$
— Proof of Lemma 11.9
have finite $(U-R)$ finite $(S i \cap R)(U-R) \cap(S i \cap R)=\{ \}$
using $U$-finite $S$-finite min-in-range $[O F\langle 0<m\rangle]$ by auto
then have sum ?c $(U-R \cup(S i \cap R))=\operatorname{sum}$ ?c $(U-R)+\operatorname{sum}$ ?c $(S i \cap$
R)
by (rule sum.union-disjoint)
moreover have $U$-split: $U-(R-S i)=U-R \cup(S i \cap R)$ using hyp(2)
by blast
moreover \{
have sum ?c $(S i \cap R)=\operatorname{card}(S i \cap R) *$ cost $R i$ by $\operatorname{simp}$
also have $\ldots=w i$ unfolding cost-def using $\langle 0<\operatorname{card}(S i \cap R)\rangle$ by simp
finally have sum ?c $(S i \cap R)=w i$.
\}
ultimately have sum ?c $(U-(R-S i))=\operatorname{sum} ? c(U-R)+w i$ by simp moreover \{
have $C \cap\{i\}=\{ \}$ using $\operatorname{hyp}(1)\langle S i \cap R \neq\{ \}\rangle$ by (auto simp: sc-def)
from sum.union-disjoint $[O F-$-this $]$ have sum $w(C \cup\{i\})=\operatorname{sum} w C+$ $w i$
using hyp(1) by (auto simp: sc-def intro: finite-subset)
\}
ultimately have 1: sum $w(C \cup\{i\})=\operatorname{sum}$ ?c $(U-(R-S i))$ - Lemma 11.9
using $\langle\operatorname{sum} w C=\operatorname{sum} c(U-R)\rangle$ by simp
have 2: $\forall i .0 \leq$ ? c $i$ using $\langle\forall i .0 \leq c i\rangle$ cost-nonneg by simp

- Proof of Lemma 11.10
have 3: $\forall k \in\{1 . . m\}$. sum ?c $(S k \cap(U-(R-S i)))$

$$
\leq\left(\sum j=\operatorname{card}(S k \cap(R-S i))+1 . . \operatorname{card}(S k) . \text { inverse } j\right) * w k
$$

## proof

fix $k$ assume $k \in\{1 . . m\}$
let ? rem $=S k \cap R-$ Remaining elements to be covered
let ? add $=S k \cap S i \cap R$ - Elements that will be covered in this step
let ? cov $=S k \cap(U-R)$ - Covered elements

- Transforming left and right sides
have sum ?c $(S k \cap(U-(R-S i)))=\operatorname{sum}$ ?c $(S k \cap(U-R \cup(S i \cap$ R))
unfolding $U$-split ..
also have $\ldots=$ sum ?c $(? \operatorname{cov} \cup ? a d d)$
by (simp add: Int-Un-distrib Int-assoc)
also have $\ldots=$ sum ?c ? cov + sum ?c ? add
by (rule sum.union-disjoint) (insert $S$-finite $\langle k \in-\rangle$, auto)
finally have lhs:

```
    sum ?c \((S k \cap(U-(R-S i)))=\) sum ?c ?cov + sum ?c ?add.
```

    have \(S k \cap(R-S i)=\) ? rem - ? add by blast
    then have card \((S k \cap(R-S i))=\operatorname{card}(\) ?rem - ?add \()\) by simp
    also have \(\ldots=\) card ? rem - card ? add
        using \(S\)-finite \(\langle k \in->\) by (auto intro: card-Diff-subset)
    finally have \(r h s\) :
        \(\operatorname{card}(S k \cap(R-S i))+1=\) card ?rem - card ?add +1 by simp
    - The apparent complexity of the remaining proof is deceiving. Much of this is just about convincing Isabelle that these sum transformations are allowed.
have sum ?c ? add $=$ card ? add $*$ cost $R$ i by $\operatorname{simp}$
also have $\ldots \leq$ card ? add $*$ cost $R k$
proof (cases ?rem $=\{ \}$ )
case True
then have card ?add $=0$ by (auto simp: card-eq- 0 -iff)
then show ?thesis by simp


## next

case False
from min-correct $[O F\langle k \in->$ this $]$ have cost $R i \leq \operatorname{cost} R k$ by simp
then show ?thesis by (simp add: mult-left-mono)
qed
also have $\ldots=$ card ? add $*$ inverse $($ card ? rem) $* w k$
by (simp add: cost-def divide-inverse-commute)
also have $\ldots=\left(\sum j \in\{\right.$ card ? rem - card ? add +1 .. card ?rem $\}$. inverse (card ?rem)) * $w k$
proof -
have card ? add $\leq$ card ?rem
using $S$-finite $\langle k \in-\rangle$ by (blast intro: card-mono)
then show ?thesis by (simp add: sum-distrib-left)
qed
also have $\ldots \leq\left(\sum j \in\{\right.$ card ? rem - card ? add +1 .. card ?rem $\}$. inverse $\left.j\right)$ * $w k$
proof -
have $\forall j \in\{$ card ?rem - card ? add +1 .. card ?rem $\}$. inverse (card ?rem) $\leq$ inverse $j$
by force
then have $\left(\sum j \in\{\right.$ card ?rem - card ?add +1 .. card ?rem $\}$. inverse (card ?rem))
$\leq\left(\sum j \in\{\right.$ card ?rem - card ?add +1 .. card ?rem $\}$. inverse $\left.j\right)$
by (blast intro: sum-mono)
with w-nonneg show ?thesis by (blast intro: mult-right-mono)
qed
finally have sum ?c ?add

$$
\leq\left(\sum j \in\{\text { card ?rem }- \text { card ?add }+1 . . \text { card ?rem }\} . \text { inverse } j\right) * w
$$

$k$.
moreover from SUM have sum ?c ?cov
$\leq\left(\sum j \in\{\right.$ card ? rem +1 .. card $(S k)\}$. inverse $\left.j\right) * w k$
using $\langle k \in\{1$.. $m\}\rangle$ by $\operatorname{simp}$
ultimately have sum ?c $(S k \cap(U-(R-S i)))$

$$
\begin{aligned}
\leq & \left(\left(\sum_{j \in\{\text { card ?rem }- \text { card ?add }+1 . . \text { card ?rem }\} . \text { inverse } j)+}^{\left.\left(\sum j \in\{\text { card ?rem }+1 \ldots \text { card }(S k)\} . \text { inverse } j\right)\right) * w k}\right.\right.
\end{aligned}
$$

unfolding lhs by argo
also have $\ldots=\left(\sum j \in\{\right.$ card ? rem - card ? add +1 .. card $(S k)\}$. inverse
j) $* w k$
proof -
have sum-split: $b \in\{a . . c\} \Longrightarrow \operatorname{sum} f\{a . . c\}=\operatorname{sum} f\{a . . b\}+\operatorname{sum} f$
$\{$ Suc $b . . c\}$
for $f::$ nat $\Rightarrow$ real and $a b c::$ nat
proof -
assume $b \in\{a . . c\}$
then have $\{a . . b\} \cup\{S u c b . . c\}=\{a . . c\}$ by force
moreover have $\{a . . b\} \cap\{$ Suc $b \ldots c\}=\{ \}$
using $\langle b \in\{a . . c\}\rangle$ by auto
ultimately show ?thesis by (metis finite-atLeastAtMost sum.union-disjoint)
qed
have $\left(\sum j \in\{\right.$ card ? rem - card ? add +1 .. card $(S k)\}$. inverse $\left.j\right)$
$=\left(\sum j \in\{\right.$ card ?rem - card ?add +1 .. card ? rem $\}$. inverse $\left.j\right)$
$+\left(\sum j \in\{\right.$ card ? rem +1 .. card $(S k)\}$. inverse $\left.j\right)$
proof (cases «?add $=\{ \}\rangle$ )
case False
then have $0<$ card ?add $0<$ card ?rem
using $S$-finite $\langle k \in \rightarrow$ by fastforce +
then have Suc (card ?rem - card ?add) $\leq$ card ?rem by simp
moreover have card ?rem $\leq \operatorname{card}(S k)$
using $S$-finite $\langle k \in->$ by (simp add: card-mono)
ultimately show ?thesis by (auto intro: sum-split)
qed simp
then show?thesis by algebra
qed
finally show sum ?c $(S k \cap(U-(R-S i)))$
$\leq\left(\sum j \in\{\operatorname{card}(S k \cap(R-S i))+1 . . \operatorname{card}(S k)\}\right.$. inverse $\left.j\right) * w k$
unfolding rhs.
qed
from 123 show ?case by blast
qed
qed
lemma cover-sum:
fixes $c::{ }^{\prime} a \Rightarrow$ real
assumes sc $C V \forall i .0 \leq c i$
shows sum c $V \leq\left(\sum i \in C\right.$.sum c $\left.(S i)\right)$
proof -
from $\operatorname{assms}(1)$ have finite $C$ by (auto simp: sc-def finite-subset)
then show ?thesis using assms (1)
proof (induction C arbitrary: V rule: finite-induct)
case (insert $i C$ )

```
    have V-split: (U (S'insert i C)) =(U (S'C)) \cupS i by auto
    have finite: finite (U (S'C)) finite (S i)
    using insert S-finite by (auto simp: sc-def)
    have sum c (S i) - sumc(\bigcup (S'C)\capSi)\leq\operatorname{sumc}(Si)
        using assms(2) by (simp add: sum-nonneg)
    then have sum c(U(S'insert iC))\leqsum c(U(S'C))+ sum c (S i)
        unfolding V-split using sum-Un[OF finite, of c] by linarith
    moreover have (\sumi\ininsert i C. sum c (Si)) = (\sumi\inC.sum c (Si)) +
sum c (Si)
        by (simp add: insert.hyps)
    ultimately show ?case using insert by (fastforce simp: sc-def)
    qed (simp add: sc-def)
qed
abbreviation H :: nat => real where H \equiv harm
definition d-star :: nat ( }\mp@subsup{d}{}{*})\mathrm{ where d}\mp@subsup{d}{}{*}\equiv\operatorname{Max}(\operatorname{card}`(S'{1..m})
lemma set-cover-bound:
    assumes inv C {} sc C' U
    shows sum wC\leqH d
proof -
    from invD(3)[OF assms(1)] obtain c where
        sum w C = sum c U \foralli. O \leqci and H-bound:
        \forallk\in{1..m}.sum c (Sk)\leqH(card (Sk))*wk-Lemma 11.10
        by (auto simp: harm-def Int-absorb2 S-subset)
    have }\forallk\in{1..m}.card (Sk)\leqd* by (auto simp:d-star-def
    then have }\forallk\in{1..m}.H(card (Sk))\leqH d by (auto simp: harm-def intro!::
sum-mono2)
    with H-bound have }\forallk\in{1..m}. sum c (Sk)\leqH d* * w
        by (metis atLeastAtMost-iff atLeastatMost-empty-iff empty-iff mult-right-mono
w-nonneg)
    moreover have C'\subseteq{1..m} using assms(2) by (simp add: sc-def)
    ultimately have }\foralli\in\mp@subsup{C}{}{\prime}\mathrm{ . sum c (Si) < H d}\mp@subsup{d}{}{*}*wi\mathrm{ by blast
    then have (\sumi\inC'. sum c (Si)) \leqH d
            by (auto simp: sum-distrib-left intro: sum-mono)
    have sum w C = sum c U by fact - Lemma 11.9
    also have ... \leq(\sumi\inC'. sum c (S i)) by (rule cover-sum[OF assms(2)]) fact
    also have ... \leqH d* * sum w C' by fact
    finally show ?thesis.
qed
theorem set-cover-approx:
VARS (R :: 'a set) (C :: nat set) ( }i\mathrm{ :: nat)
    {True}
    R:=U;C:={};
```

```
    WHILE R\not={} INV {inv C R} DO
    i:= min-arg R m;
    R:=R-Si;
    C:=C\cup{i}
    OD
    {scC U\wedge(\forall\mp@subsup{C}{}{\prime}.sc C'}U\longrightarrow\operatorname{sum}wC\leqH\mp@subsup{d}{}{*}*\operatorname{sum}w\mp@subsup{C}{}{\prime})
proof (vcg, goal-cases)
    case 1 show ?case by (rule inv-init)
next
    case 2 thus ?case using inv-step ..
next
    case (3 R C i)
    then have sc C U unfolding inv-def by auto
    with 3 show ?case by (auto intro: set-cover-bound)
qed
end
end
```


## 3 Independent Set

theory Approx-MIS-Hoare
imports
HOL-Hoare.Hoare-Logic
HOL-Library.Disjoint-Sets
begin
The algorithm is classical, the proofs are inspired by the ones by Berghammer and Müller-Olm [1]. In particular the approximation ratio is improved from $\Delta+1$ to $\Delta$.

### 3.1 Graph

A set set is simply a set of edges, where an edge is a 2-element set.
definition independent-vertices :: 'a set set $\Rightarrow$ 'a set $\Rightarrow$ bool where
independent-vertices $E S \longleftrightarrow S \subseteq \bigcup E \wedge(\forall v 1$ v2. v1 $\in S \wedge v 2 \in S \longrightarrow\{v 1, v 2\}$ $\notin E)$
locale Graph-E =
fixes $E$ :: 'a set set
assumes finite- $E$ : finite $E$
assumes edges2: $e \in E \Longrightarrow$ card $e=2$
begin
fun vertices :: 'a set set $\Rightarrow$ ' $a$ set where vertices $G=\bigcup G$
abbreviation $V$ :: 'a set where

$$
V \equiv \text { vertices } E
$$

definition approximation-miv :: nat $\Rightarrow$ 'a set $\Rightarrow$ bool where
approximation-miv $n S \longleftrightarrow$ independent-vertices $E S \wedge\left(\forall S^{\prime}\right.$. independent-vertices $\left.E S^{\prime} \longrightarrow \operatorname{card} S^{\prime} \leq \operatorname{card} S * n\right)$
fun neighbors :: ' $a \Rightarrow$ 'a set where
neighbors $v=\{u .\{u, v\} \in E\}$
fun degree-vertex :: ' $a \Rightarrow$ nat where
degree-vertex $v=$ card (neighbors $v$ )
abbreviation $\Delta::$ nat where
$\Delta \equiv \operatorname{Max}\{$ degree-vertex $u \mid u . u \in V\}$
lemma finite-edges: $e \in E \Longrightarrow$ finite $e$
using card-ge-0-finite and edges2 by force
lemma finite- $V$ : finite $V$
using finite-edges and finite- $E$ by auto
lemma finite-neighbors: finite (neighbors u)
using finite- $V$ and rev-finite-subset $[o f ~ V$ neighbors $u]$ by auto
lemma independent-vertices-finite: independent-vertices $E S \Longrightarrow$ finite $S$
by (metis rev-finite-subset independent-vertices-def vertices.simps finite- $V$ )

```
lemma edge-ex-vertices: \(e \in E \Longrightarrow \exists u v . u \neq v \wedge e=\{u, v\}\)
proof -
    assume \(e \in E\)
    then have card \(e=\) Suc (Suc 0) using edges2 by auto
    then show \(\exists u v . u \neq v \wedge e=\{u, v\}\)
        by (metis card-eq-SucD insertI1)
qed
lemma \(\Delta\)-pos \([\) simp \(]: E=\{ \} \vee 0<\Delta\)
proof cases
    assume \(E=\{ \}\)
    then show \(E=\{ \} \vee 0<\Delta\) by auto
next
    assume \(1: E \neq\{ \}\)
    then have \(V \neq\{ \}\) using edges2 by fastforce
    moreover have finite \(\{\) degree-vertex \(u \mid u . u \in V\}\)
        by (metis finite-V finite-imageI Setcompr-eq-image)
    ultimately have 2: \(\Delta \in\{\) degree-vertex \(u \mid u . u \in V\}\) using Max-in by auto
    have \(\Delta \neq 0\)
    proof
        assume \(\Delta=0\)
```

with 2 obtain $u$ where $3: u \in V$ and 4: degree-vertex $u=0$ by auto
from 3 obtain $e$ where $5: e \in E$ and $u \in e$ by auto
moreover with 4 have $\forall v .\{u, v\} \neq e$ using finite-neighbors insert-absorb
by fastforce
ultimately show False using edge-ex-vertices by auto
qed
then show $E=\{ \} \vee 0<\Delta$ by auto
qed

```
lemma \Delta-max-degree: }u\inV\Longrightarrow\mathrm{ degree-vertex }u\leq
proof -
    assume H:u\inV
    have finite {degree-vertex u |u.u\inV}
    by (metis finite-V finite-imageI Setcompr-eq-image)
    with H show degree-vertex }u\leq\Delta\mathrm{ using Max-ge by auto
qed
```


### 3.2 Wei's algorithm: $(\Delta+1)$-approximation

The 'functional' part of the invariant, used to prove that the algorithm produces an independent set of vertices.

```
definition inv-iv :: 'a set \(\Rightarrow\) 'a set \(\Rightarrow\) bool where
inv-iv \(S X \longleftrightarrow\) independent-vertices \(E S\)
    \(\wedge X \subseteq V\)
    \(\wedge(\forall v 1 \in(V-X) . \forall v 2 \in S .\{v 1, v 2\} \notin E)\)
    \(\wedge S \subseteq X\)
```

Strenghten the invariant with an approximation ratio $r$ :
definition inv-approx $::$ 'a set $\Rightarrow$ 'a set $\Rightarrow$ nat $\Rightarrow$ bool where inv-approx $S X r \longleftrightarrow$ inv-iv $S X \wedge$ card $X \leq$ card $S * r$

Preservation of the functional invariant:

```
lemma inv-preserv:
    fixes \(S\) :: ' \(a\) set
        and \(X\) :: ' \(a\) set
        and \(x::\) ' \(a\)
    assumes inv: inv-iv \(S X\)
            and \(x\)-def: \(x \in V-X\)
        shows inv-iv (insert \(x S)(X \cup\) neighbors \(x \cup\{x\})\)
proof -
    have inv1: independent-vertices \(E S\)
        and inv2: \(X \subseteq V\)
        and inv3: \(S \subseteq X\)
        and inv4: \(\forall v 1\) v2. v1 \(\in(V-X) \wedge v 2 \in S \longrightarrow\{v 1, v 2\} \notin E\)
        using inv unfolding inv-iv-def by auto
    have finite-S: finite \(S\) using inv1 and independent-vertices-finite by auto
    have \(S 1: \forall y \in S .\{x, y\} \notin E\) using inv4 and \(x\)-def by blast
    have S2: \(\forall x \in S . \forall y \in S .\{x, y\} \notin E\) using inv1 unfolding independent-vertices-def
by metis
```

```
have S3:v1 \in insert x S \Longrightarrow v2 \in insert x S \Longrightarrow {v1,v2} & E for v1 v2
    proof -
    assume v1\in insert x S and v2 \in insert x S
    then consider
        (a) v1 = x and v2 = x
        | (b) v1 = x and v2 \inS
        | (c) v1 \inS and v2 = x
        | (d) v1 \inS and v2 \inS
        by auto
    then show {v1,v2} }\not\in
    proof cases
        case a then show ?thesis using edges2 by force
    next
        case b then show ?thesis using S1 by auto
    next
        case c then show ?thesis using S1 by (metis doubleton-eq-iff)
    next
        case d then show ?thesis using S2 by auto
    qed
    qed
have independent-vertices E (insert x S)
    using S3 and inv1 and x-def unfolding independent-vertices-def by auto
moreover have }X\cup\mathrm{ neighbors }x\cup{x}\subseteq
proof
    fix }x
    assume xa \inX\cup neighbors }x\cup{x
    then consider (a) xa \inX|(b) xa \in neighbors }x|(c)xa=x by aut
    then show }xa\in
    proof cases
        case a
        then show ?thesis using inv2 by blast
    next
        case b
        then show ?thesis by auto
    next
        case c
        then show ?thesis using x-def by blast
    qed
qed
moreover have insert x S\subseteqX\cup neighbors }x\cup{x}\mathrm{ using inv3 by auto
moreover have v1\inV-(X\cup neighbors }x\cup{x})\Longrightarrowv2\in insert x S
{v1,v2} &E for v1 v2
proof -
    assume H:v1 \inV - (X\cup neighbors }x\cup{x})\mathrm{ and v2 G insert x S
    then consider (a) v2 =x |(b) v2 }\inS\mathrm{ by auto
```

```
    then show {v1,v2} &E
    proof cases
        case a
        with H have v1 & neighbors v2 by blast
        then show ?thesis by auto
    next
        case b
        from H have v1\inV - X by blast
        with b and inv4 show ?thesis by blast
    qed
qed
```

ultimately show inv-iv (insert $x S)(X \cup$ neighbors $x \cup\{x\})$ unfolding inv-iv-def by blast
qed
lemma inv-approx-preserv:
assumes inv: inv-approx $S X(\Delta+1)$
and $x$-def: $x \in V-X$
shows inv-approx $($ insert $x S)(X \cup$ neighbors $x \cup\{x\})(\Delta+1)$
proof -
have finite-S: finite $S$ using inv and independent-vertices-finite unfolding inv-approx-def inv-iv-def by auto
have $S x: x \notin S$ using inv and $x$-def unfolding inv-approx-def inv-iv-def by blast
from inv have inv-iv $S X$ unfolding inv-approx-def by auto with $x$-def have inv-iv (insert $x S)(X \cup$ neighbors $x \cup\{x\})$ proof (intro inv-preserv, auto) qed
moreover have card $(X \cup$ neighbors $x \cup\{x\}) \leq \operatorname{card}($ insert $x S) *(\Delta+1)$ proof -
have degree-vertex $x \leq \Delta$ using $\Delta$-max-degree and $x$-def by auto
then have card (neighbors $x \cup\{x\}) \leq \Delta+1$ using card-Un-le [of neighbors $x\{x\}]$ by auto
then have card $(X \cup$ neighbors $x \cup\{x\}) \leq$ card $X+\Delta+1$ using card-Un-le [of $X$ neighbors $x \cup\{x\}]$ by auto
also have $\ldots \leq \operatorname{card} S *(\Delta+1)+\Delta+1$ using inv unfolding inv-approx-def
by auto
also have $\ldots=\operatorname{card}($ insert $x S) *(\Delta+1)$ using finite- $S$ and $S x$ by auto
finally show ?thesis .
qed
ultimately show inv-approx (insert $x S)(X \cup$ neighbors $x \cup\{x\})(\Delta+1)$
unfolding inv-approx-def by auto
qed
lemma inv-approx: independent-vertices $E S \Longrightarrow$ card $V \leq \operatorname{card} S * r \Longrightarrow$ ap-

```
proximation-miv r S
proof -
    assume 1: independent-vertices E S and 2: card V \leqcard S*r
    have independent-vertices E S' \Longrightarrow card S'\leqcard S*r for }\mp@subsup{S}{}{\prime
    proof -
        assume independent-vertices E S'
        then have }\mp@subsup{S}{}{\prime}\subseteqV\mathrm{ unfolding independent-vertices-def by auto
        then have card S}\mp@subsup{S}{}{\prime}\leq\mathrm{ card V using finite-V and card-mono by auto
        also have ... \leqcard S*r using 2 by auto
        finally show card S' 
    qed
    with 1 show approximation-miv r S unfolding approximation-miv-def by auto
qed
theorem wei-approx-\Delta-plus-1:
VARS (S :: 'a set) (X :: 'a set) (x :: 'a)
    { True }
    S:= {};
    X:={};
    WHILE X = V
    INV { inv-approx SX (\Delta+1)}
    DO x := (SOME x. x \inV - X);
        S:= insert x S;
        X:=X\cup neighbors }x\cup{x
    OD
    { approximation-miv (\Delta+1)S}
proof (vcg, goal-cases)
    case (1S X x)
    then show ?case unfolding inv-approx-def inv-iv-def independent-vertices-def
by auto
next
    case (2 S X x)
    let ?x = (SOME x. x\inV - X)
    have V-X\not={} using 2 unfolding inv-approx-def inv-iv-def by blast
    then have ?}x\inV-X using some-in-eq by meti
    with 2 show ?case using inv-approx-preserv by auto
next
    case (3 S X x)
    then show ?case using inv-approx unfolding inv-approx-def inv-iv-def by auto
qed
```


### 3.3 Wei's algorithm: $\Delta$-approximation

The previous approximation uses very little information about the optimal solution (it has at most as many vertices as the set itself). With some extra effort we can improve the ratio to $\Delta$ instead of $\Delta+1$. In order to do that we must show that among the vertices removed in each iteration, at most $\Delta$ could belong to an optimal solution. This requires carrying around a set $P$
(via a ghost variable) which records the vertices deleted in each iteration.
definition inv-partition $::$ 'a set $\Rightarrow{ }^{\prime}$ 'a set $\Rightarrow$ 'a set set $\Rightarrow$ bool where
inv-partition $S X P \longleftrightarrow$ inv-iv $S X$

$$
\begin{aligned}
& \wedge \bigcup P=X \\
& \wedge(\forall p \in P \cdot \exists s \in V \cdot p=\{s\} \cup \text { neighbors } s) \\
& \wedge \text { card } P=\text { card } S \\
& \wedge \text { finite } P
\end{aligned}
$$

lemma inv-partition-preserv:
assumes inv: inv-partition $S X P$
and $x$-def: $x \in V-X$
shows inv-partition (insert $x S)(X \cup$ neighbors $x \cup\{x\})$ (insert $(\{x\} \cup$ neighbors $x$ ) $P$ )
proof -
have finite-S: finite $S$ using inv and independent-vertices-finite unfolding inv-partition-def inv-iv-def by auto
have $S x: x \notin S$ using inv and $x$-def unfolding inv-partition-def inv-iv-def by blast
from inv have inv-iv $S X$ unfolding inv-partition-def by auto with $x$-def have inv-iv (insert $x S)(X \cup$ neighbors $x \cup\{x\})$
proof (intro inv-preserv, auto) qed
moreover have $\cup($ insert $(\{x\} \cup$ neighbors $x) P)=X \cup$ neighbors $x \cup\{x\}$
using inv unfolding inv-partition-def by auto
moreover have $(\forall p \in$ insert $(\{x\} \cup$ neighbors $x) P . \exists s \in V . p=\{s\} \cup$ neighbors s)
using inv and $x$-def unfolding inv-partition-def by auto
moreover have card (insert $(\{x\} \cup$ neighbors $x) P)=\operatorname{card}($ insert $x S)$
proof -
from $x$-def and inv have $x \notin \bigcup P$ unfolding inv-partition-def by auto
then have $\{x\} \cup$ neighbors $x \notin P$ by auto
then have card (insert $(\{x\} \cup$ neighbors $x) P)=$ card $P+1$ using inv unfolding inv-partition-def by auto
moreover have card (insert $x S$ ) $=\operatorname{card} S+1$ using $S x$ and finite- $S$ by auto
ultimately show ?thesis using inv unfolding inv-partition-def by auto qed
moreover have finite (insert $(\{x\} \cup$ neighbors $x) P$ )
using inv unfolding inv-partition-def by auto
ultimately show inv-partition (insert $x S)(X \cup$ neighbors $x \cup\{x\})($ insert $(\{x\}$ $\cup$ neighbors $x$ ) $P$ )
unfolding inv-partition-def by auto
qed
lemma card-Union-le-sum-card:

```
    fixes U :: ' }a\mathrm{ set set
    assumes }\forallu\inU\mathrm{ . finite }
    shows card }(\bigcupU)\leq\mathrm{ sum card }
proof (cases finite U)
    case False
    then show card ( \bigcupU)\leq sum card U
    using card-eq-0-iff finite-UnionD by auto
next
    case True
    then show card ( UU)\leq sum card U
    proof (induct U rule: finite-induct)
        case empty
        then show ?case by auto
    next
        case (insert x F)
        then have card (U(insert x F)) \leq card(x) + card (UF) using card-Un-le by
auto
    also have ... \leq card (x) + sum card F using insert.hyps by auto
    also have ... = sum card (insert x F) using sum.insert-if and insert.hyps by
auto
    finally show ?case .
    qed
qed
lemma sum-card:
    fixes U :: 'a set set
        and n :: nat
    assumes }\forallS\inU.card S\leqn
    shows sum card U\leqcard U*n
proof cases
    assume infinite }U\veeU={
    then have sum card U = 0 using sum.infinite by auto
    then show sum card U\leqcard U*n by auto
next
    assume }\neg(\mathrm{ infinite }U\veeU={}
    with assms have finite U and U\not={}\mathrm{ and }\forallS\inU.card S \leqn by auto
    then show sum card U\leq card U*n
    proof (induct U rule: finite-ne-induct)
        case (singleton x)
        then show ?case by auto
    next
        case (insert x F)
        assume }\forallS\in\mathrm{ insert x F.card S 
        then have 1:card }x\leqn\mathrm{ and 2:sum card F}\leq\operatorname{card F*n using insert.hyps
by auto
    then have sum card (insert x F) = card }x+\mathrm{ sum card F using sum.insert-if
and insert.hyps by auto
    also have ... \leqn+card F*n using 1 and 2 by auto
```

```
    also have ... = card (insert x F)*n using card-insert-if and insert.hyps by
auto
    finally show ?case .
    qed
qed
```

lemma $x$-or-neighbors:
fixes $P$ :: ' $a$ set set
and $S::$ ' $a$ set
assumes inv: $\forall p \in P . \exists s \in V . p=\{s\} \cup$ neighbors $s$
and ivS: independent-vertices $E S$
shows $\forall p \in P$. card $(S \cap p) \leq \Delta$
proof
fix $p$
assume $p \in P$
then obtain $s$ where $1: s \in V \wedge p=\{s\} \cup$ neighbors $s$ using inv by blast
then show card $(S \cap p) \leq \Delta$
proof cases
assume $s \in S$
then have $S \cap$ neighbors $s=\{ \}$ using ivS unfolding independent-vertices-def
by auto
then have $S \cap p \subseteq\{s\}$ using 1 by auto
then have 2: card $(S \cap p) \leq 1$ using subset-singletonD by fastforce
consider (a) $E=\{ \} \mid(b) 0<\Delta$ using $\Delta$-pos by auto
then show card $(S \cap p) \leq \Delta$
proof cases
case $a$
then have $S=\{ \}$ using ivS unfolding independent-vertices-def by auto
then show ?thesis by auto
next
case $b$
then show ?thesis using 2 by auto
qed
next
assume $s \notin S$
with 1 have $S \cap p \subseteq$ neighbors s by auto
then have card $(S \cap p) \leq$ degree-vertex $s$ using card-mono and finite-neighbors
by auto
then show card $(S \cap p) \leq \Delta$ using 1 and $\Delta$-max-degree [of $s$ ] by auto
qed
qed
lemma inv-partition-approx: inv-partition $S V P \Longrightarrow$ approximation-miv $\Delta S$ proof -
assume H1: inv-partition $S V P$
then have independent-vertices $E S$ unfolding inv-partition-def inv-iv-def by auto
moreover have independent-vertices $E S^{\prime} \Longrightarrow \operatorname{card} S^{\prime} \leq \operatorname{card} S * \Delta$ for $S^{\prime}$ proof -
let $? I=\left\{S^{\prime} \cap p \mid p . p \in P\right\}$
assume H2: independent-vertices $E S^{\prime}$
then have $S^{\prime} \subseteq V$ unfolding independent-vertices-def using vertices.simps
by blast
with $H 1$ have $S^{\prime}=S^{\prime} \cap \bigcup P$ unfolding inv-partition-def by auto
then have $S^{\prime}=\left(\bigcup p \in P . S^{\prime} \cap p\right)$ using Int-Union by auto
then have $S^{\prime}=\bigcup$ ?I by blast
moreover have finite $S^{\prime}$ using H2 and independent-vertices-finite by auto
then have $p \in P \Longrightarrow$ finite ( $S^{\prime} \cap p$ ) for $p$ by auto
ultimately have card $S^{\prime} \leq$ sum card ?I using card-Union-le-sum-card [of ?I]
by auto
also have $\ldots \leq$ card ? $I * \Delta$
using $x$-or-neighbors [of P S ]
and sum-card [of ?I $\Delta$ ]
and H1 and H2 unfolding inv-partition-def by auto
also have $\ldots \leq \operatorname{card} P * \Delta$
proof -
have finite $P$ using H1 unfolding inv-partition-def by auto
then have card $? I \leq$ card $P$
using Setcompr-eq-image [of $\lambda p . S^{\prime} \cap p P$ ]
and card-image-le unfolding inv-partition-def by auto
then show? ?hesis by auto
qed
also have $\ldots=$ card $S * \Delta$ using $H 1$ unfolding inv-partition-def by auto
ultimately show card $S^{\prime} \leq \operatorname{card} S * \Delta$ by auto
qed
ultimately show approximation-miv $\Delta S$ unfolding approximation-miv-def by auto
qed
theorem wei-approx- $\Delta$ :
$\operatorname{VARS}\left(S::{ }^{\prime} a\right.$ set $)\left(X::{ }^{\prime} a \operatorname{set}\right)\left(x::^{\prime} a\right)$
\{ True \}
$S:=\{ \} ;$
$X:=\{ \} ;$
WHILE $X \neq V$
INV $\{\exists P$. inv-partition $S X P\}$
DO $x:=(S O M E$ x. $x \in V-X)$;
$S:=$ insert $x S$;
$X:=X \cup$ neighbors $x \cup\{x\}$
OD
\{ approximation-miv $\Delta S$ \}
proof (vcg, goal-cases)
case (1SXx)
have inv-partition $\}\}\}$ unfolding inv-partition-def inv-iv-def independent-vertices-def
by auto
then show? case by auto
next
case (2 $S X x$ )
let $? x=($ SOME $x . x \in V-X)$
from 2 obtain $P$ where $I$ : inv-partition $S X P$ by auto
then have $V-X \neq\{ \}$ using 2 unfolding inv-partition-def by auto
then have $? x \in V-X$ using some-in-eq by metis
with $I$ have inv-partition (insert ? $x S)(X \cup$ neighbors ? $x \cup\{? x\})$ (insert $(\{? x\}$
$\cup$ neighbors ? $x$ ) $P$ )
using inv-partition-preserv by blast
then show ?case by auto
next
case (3 $S X x$ )
then show ?case using inv-partition-approx unfolding inv-approx-def by auto qed

### 3.4 Wei's algorithm with dynamically computed approximation ratio

In this subsection, we augment the algorithm with a variable used to compute the effective approximation ratio of the solution. In addition, the vertex of smallest degree is picked. With this heuristic, the algorithm achieves an approximation ratio of $(\Delta+2) / 3$, but this is not proved here.
definition vertex-heuristic :: 'a set $\Rightarrow{ }^{\prime} a \Rightarrow$ bool where
vertex-heuristic $X v=(\forall u \in V-X$. card (neighbors $v-X) \leq$ card (neighbors $u-X)$ )
lemma ex-min-finite-set:
fixes $S$ :: 'a set and $f::{ }^{\prime} a \Rightarrow n a t$
shows finite $S \Longrightarrow S \neq\{ \} \Longrightarrow \exists x . x \in S \wedge(\forall y \in S . f x \leq f y)$
$($ is ? P1 $\Longrightarrow ? P 2 \Longrightarrow \exists x$. ? $\operatorname{minf} S x)$
proof (induct $S$ rule: finite-ne-induct)
case (singleton $x$ )
have ? $\operatorname{minf}\{x\} x$ by auto
then show? case by auto
next
case (insert $x$ F)
from insert(4) obtain $y$ where Py: ? minf $F y$ by auto
show $\exists z$. ?minf (insert $x F) z$
proof cases
assume $f x<f y$
then have ? minf (insert $x F$ ) $x$ using Py by auto
then show? case by auto

```
    next
        assume \(\neg f x<f y\)
        then have ? minf (insert \(x F\) ) y using Py by auto
        then show? case by auto
    qed
qed
lemma inv-approx-preserv2:
    fixes \(S\) :: ' \(a\) set
        and \(X::\) ' \(a\) set
        and \(s::\) nat
        and \(x::{ }^{\prime} a\)
    assumes inv: inv-approx \(S X s\)
        and \(x\)-def: \(x \in V-X\)
        shows inv-approx (insert \(x S\) ) ( \(X \cup\) neighbors \(x \cup\{x\}\) ) (max (card (neighbors
\(x \cup\{x\}-X)) s\) )
proof -
    have finite- \(S\) : finite \(S\) using inv and independent-vertices-finite unfolding
inv-approx-def inv-iv-def by auto
    have \(S x: x \notin S\) using inv and \(x\)-def unfolding inv-approx-def inv-iv-def by
blast
    from inv have inv-iv \(S X\) unfolding inv-approx-def by auto
    with \(x\)-def have inv-iv (insert \(x S)(X \cup\) neighbors \(x \cup\{x\})\)
    proof (intro inv-preserv, auto) qed
    moreover have card \((X \cup\) neighbors \(x \cup\{x\}) \leq \operatorname{card}\) (insert \(x S) * \max (\) card
(neighbors \(x \cup\{x\}-X)) s\)
    proof -
    let ? \(N=\) neighbors \(x \cup\{x\}-X\)
    have card \((X \cup ? N) \leq\) card \(X+\) card ? \(N\) using card-Un-le \([o f X\) ? \(N\) ] by auto
    also have \(\ldots \leq\) card \(S * s+\) card \(? N\) using inv unfolding inv-approx-def by
auto
    also have \(\ldots \leq\) card \(S * \max (\) card ? \(N\) ) \(s+\) card ? \(N\) by auto
    also have \(\ldots \leq \operatorname{card} S * \max (\) card ? \(N) s+\max (\) card ?N) \(s\) by auto
    also have \(\ldots=\operatorname{card}(\) insert \(x S) * \max (\operatorname{card} ? N) s\) using \(S x\) and finite- \(S\) by
auto
    finally show ?thesis by auto
    qed
    ultimately show inv-approx (insert \(x S)(X \cup\) neighbors \(x \cup\{x\})\) (max (card
(neighbors \(x \cup\{x\}-X)\) ) s)
    unfolding inv-approx-def by auto
qed
theorem wei-approx-min-degree-heuristic:
\(\operatorname{VARS}\left(S::{ }^{\prime} a \operatorname{set}\right)\left(X::{ }^{\prime} a \operatorname{set}\right)\left(x::{ }^{\prime} a\right)(r::\) nat \()\)
    \{ True \}
    \(S:=\{ \} ;\)
```

```
    X:= {};
    r:= 0;
    WHILE X }\not=
    INV { inv-approx S X r }
    DO x := (SOME x. x \inV - X ^ vertex-heuristic X x);
        S:= insert x S;
        r:= max (card (neighbors }x\cup{x}-X))r
        X:= X\cup neighbors }x\cup{x
    OD
    { approximation-miv r S }
proof (vcg, goal-cases)
    case (1 S X x r)
    then show ?case unfolding inv-approx-def inv-iv-def independent-vertices-def
by auto
next
    case (2 S X x r)
    let ?}\cdotx=(SOME x. x\inV-X^ vertex-heuristic X x)
    have V-X\not={} using 2 unfolding inv-approx-def inv-iv-def by blast
    moreover have finite ( }V-X\mathrm{ ) using 2 and finite- V by auto
    ultimately have }\existsx.x\inV-X\wedge vertex-heuristic X x
        using ex-min-finite-set [where ?f = \lambdax.card (neighbors }x-X)
        unfolding vertex-heuristic-def by auto
    then have x-def:?x }\inV-X\wedge\mathrm{ vertex-heuristic }X\mathrm{ ? ? 
    using someI-ex [where ?P = \lambdax. x G V - X ^ vertex-heuristic X x] by auto
    with 2 show ?case using inv-approx-preserv2 by auto
next
    case (3 S X x r)
    then show ?case using inv-approx unfolding inv-approx-def inv-iv-def by auto
qed
end
end
```


## 4 Load Balancing

```
theory Approx-LB-Hoare
    imports Complex-Main HOL-Hoare.Hoare-Logic
begin
```

This is a formalization of the load balancing algorithms and proofs in the book by Kleinberg and Tardos [4].
hide-const (open) sorted

```
lemma sum-le-card-Max: \(\llbracket\) finite \(A ; A \neq\{ \} \rrbracket \Longrightarrow \operatorname{sum} f A \leq \operatorname{card} A * M a x(f\) '
A)
proof \((\) induction \(A\) rule: finite-ne-induct)
```

```
    case (singleton x)
    then show ?case by simp
next
    case (insert x F)
    then show ?case by (auto simp: max-def order.trans[of sum f F card F * Max
(f`}F)]
qed
lemma Max-const[simp]:\llbracket finite A;A\not={}\rrbracket\Longrightarrow Max ((\lambda-.c)'A)=c
using Max-in image-is-empty by blast
abbreviation Max0 :: nat set }=>\mathrm{ nat where
Max }N\equiv(\mathrm{ if }N={} then 0 else Max N
fun f-Max0 :: (nat => nat) => nat => nat where
    f-Max f 0 = 0
|-Max0 f (Suc x) = max (f(Suc x)) (f-Max0 f x )
```



```
    by (induction n) (auto simp: not-le atLeastAtMostSuc-conv)
lemma f-Max 
    \forallx\in{1..m}.T }x\leqf-Ma\mp@subsup{x}{0}{}T
    m>0\Longrightarrow\existsx\in{1..m}.Tx=f-Max T Tm
    apply (induction m)
        apply simp-all
    apply (metis atLeastAtMost-iff le-Suc-eq max.cobounded1 max.coboundedI2)
subgoal for m by (cases <m=0`) (auto simp: max-def)
done
lemma f-Max0-mono:
    y \leqTx\Longrightarrowf-Max }(T(x:=y))m\leqf-Max0 T m
    Tx\leqy\Longrightarrowf-Max0 T m \leqf-Max0 (T (x:=y)) m
    by (induction m) auto
lemma f-Max -out-of-range [simp]:
    x\not\in{1..k}\Longrightarrowf-Max (T (x:=y))k=f-Max0}T
    by (induction k) auto
lemma fun-upd-f-Max0:
    assumes }x\in{1..m}Tx\leq
    shows f-Max0 (T (x:=y)) m= max y (f-Max0 T m)
    using assms by (induction m) auto
locale LoadBalancing =
    fixes t:: nat => nat
        and m :: nat
        and n :: nat
    assumes m-gt-0:m>0
```


## begin

### 4.1 Formalization of a Correct Load Balancing

### 4.1.1 Definition

definition $l b::($ nat $\Rightarrow$ nat $) \Rightarrow($ nat $\Rightarrow$ nat set $) \Rightarrow$ nat $\Rightarrow$ bool where
$l b T A j=((\forall x \in\{1 . . m\} . \forall y \in\{1 . . m\} . x \neq y \longrightarrow A x \cap A y=\{ \})-$ No job is assigned to more than one machine
$\wedge(\bigcup x \in\{1 . . m\} . A x)=\{1 . . j\}$ - Every job is assigned
$\wedge\left(\forall x \in\{1 . . m\} .\left(\sum j \in A x . t j\right)=T x\right)$ — The processing times sum up to the correct load)
abbreviation makespan $::($ nat $\Rightarrow$ nat $) \Rightarrow$ nat where
makespan $T \equiv f-$ Max $_{0} T m$
lemma makespan-def': makespan $T=\operatorname{Max}(T$ ' $\{1 . . m\})$
using $m$-gt- 0 by (simp add: $f$-Max $x_{0}$-equiv)
lemma makespan-correct:

$$
\forall x \in\{1 . . m\} . T x \leq \text { makespan } T
$$

$\exists x \in\{1 . . m\} . T x=$ makespan $T$
using $f$-Max $0_{0}$-correct $m$-gt- 0 by auto
lemma $l b E$ :
assumes $l b T A j$
shows $\forall x \in\{1 . . m\} . \forall y \in\{1 . . m\} . x \neq y \longrightarrow A x \cap A y=\{ \}$
$(\bigcup x \in\{1 . . m\} . A x)=\{1 . . j\}$
$\forall x \in\{1 . . m\} .\left(\sum y \in A x . t y\right)=T x$
using assms unfolding $l b$-def by blast+
lemma $l b I$ :
assumes $\forall x \in\{1 . . m\} . \forall y \in\{1 . . m\} . x \neq y \longrightarrow A x \cap A y=\{ \}$

$$
(\bigcup x \in\{1 . . m\} . A x)=\{1 . . j\}
$$

$\forall x \in\{1 . . m\} .\left(\sum y \in A x . t y\right)=T x$
shows $l b T A j$ using assms unfolding lb-def by blast
lemma $A$-lb-finite $[$ simp]:
assumes $l b T A j x \in\{1 . . m\}$
shows finite $\left(\begin{array}{ll}A & x\end{array}\right)$
by (metis lbE(2) assms finite-UN finite-atLeastAtMost)
If $A x$ is pairwise disjoint for all $x \in\{1 . . m\}$, then the the sum over the sums of the individual $A x$ is equal to the sum over the union of all $A x$.

```
lemma sum-sum-eq-sum-Un:
    fixes \(A::\) nat \(\Rightarrow\) nat set
    assumes \(\forall x \in\{1 . . m\} . \forall y \in\{1 . . m\} . x \neq y \longrightarrow A x \cap A y=\{ \}\)
        and \(\forall x \in\{1 . . m\}\). finite \((A x)\)
    shows \(\left(\sum x \in\{1 . . m\} .\left(\sum y \in A x . t y\right)\right)=\left(\sum x \in(\bigcup y \in\{1 . . m\} . A y) . t x\right)\)
```

```
    using assms
proof (induction m)
    case (Suc m)
    have FINITE: finite ( }\bigcupx\in{1..m}.A x) finite (A (Suc m)
        using Suc.prems(2) by auto
    have }\forallx\in{1..m}.Ax\capA(Sucm)={
        using Suc.prems(1) by simp
    then have DISJNT: (\bigcupx\in{1..m}.A x) \cap(A(Suc m))={} using Union-disjoint
by blast
    have (\sumx\in(\bigcupy\in{1..m}.A y).t x) +(\sumx\inA(Suc m).tx)
            =(\sumx\in((\bigcupy\in{1..m}.A y)\cupA(Suc m)).tx)
        using sum.union-disjoint[OF FINITE DISJNT, symmetric].
    also have ... = (\sumx\in(\bigcupy\in{1..Suc m}.A y).t x)
    by (metis UN-insert image-Suc-lessThan image-insert inf-sup-aci(5) lessThan-Suc)
    finally show ?case using Suc by auto
qed simp
```

If $T$ and $A$ are a correct load balancing for $j$ jobs and $m$ machines, then the sum of the loads has to be equal to the sum of the processing times of the jobs

```
lemma lb-impl-job-sum:
    assumes \(l b T A j\)
    shows \(\left(\sum x \in\{1 . . m\} . T x\right)=\left(\sum x \in\{1 . . j\} . t x\right)\)
proof -
    note \(l b r u l e s=l b E[O F\) assms \(]\)
    from assms have FINITE: \(\forall x \in\{1 . . m\}\). finite \((A x)\) by simp
    have \(\left(\sum x \in\{1 . . m\} . T x\right)=\left(\sum x \in\{1 . . m\} .\left(\sum y \in A x . t y\right)\right)\)
        using lbrules (3) by simp
    also have \(\ldots=\left(\sum x \in\{1 . . j\} . t x\right)\)
        using sum-sum-eq-sum-Un[OF lbrules(1) FINITE]
        unfolding lbrules(2).
    finally show ?thesis .
qed
```


### 4.1.2 Lower Bounds for the Makespan

If $T$ and $A$ are a correct load balancing for $j$ jobs and $m$ machines, then the processing time of any job $x \in\{1 . . j\}$ is a lower bound for the load of some machine $y \in\{1 . . m\}$
lemma job-lower-bound-machine:
assumes $l b T A j x \in\{1 . . j\}$
shows $\exists y \in\{1 . . m\} . t x \leq T y$
proof -
note lbrules $=l b E[$ OF assms(1)]
have $\exists y \in\{1 . . m\} . x \in A y$ using lbrules(2) assms(2) by blast
then obtain $y$ where $y$-def: $y \in\{1 . . m\} x \in A y .$.
moreover have finite ( $A$ y) using $\operatorname{assms}(1) y$-def(1) by simp
ultimately have $t x \leq\left(\sum x \in A\right.$. $\left.t x\right)$ using lbrules(1) member-le-sum by fast
also have $\ldots=T y$ using $\operatorname{lbrules}(3) y$ - $d e f(1)$ by blast
finally show ?thesis using $y$-def(1) by blast
qed
As the load of any machine is a lower bound for the makespan, the processing time of any job $x \in\{1 . . j\}$ has to also be a lower bound for the makespan. Follows from job-lower-bound-machine and makespan-correct.
lemma job-lower-bound-makespan:
assumes $l b T A j x \in\{1 . . j\}$
shows $t x \leq$ makespan $T$
by (meson job-lower-bound-machine [OF assms] makespan-correct(1) le-trans)
The makespan over $j$ jobs is a lower bound for the makespan of any correct load balancing for $j$ jobs.

```
lemma max-job-lower-bound-makespan:
    assumes lb TAj
    shows Max0 (t'{1..j})\leqmakespan T
    using job-lower-bound-makespan[OF assms] by fastforce
lemma job-dist-lower-bound-makespan:
    assumes lb TAj
    shows (\sumx\in{1..j}.tx)/m\leq makespan T
proof -
    have (\sumx\in{1..j}.t }\\mathrm{ ) }\leqm*\mathrm{ makespan T
        using assms lb-impl-job-sum[symmetric]
            and sum-le-card-Max[of {1..m}] m-gt-0 by (simp add: makespan-def')
    then have real (\sumx\in{1..j}.t t ) \leqreal m*real (makespan T)
        using of-nat-mono by fastforce
    then show ?thesis by (simp add: field-simps m-gt-0)
qed
```


### 4.2 The Greedy Approximation Algorithm

This function will perform a linear scan from $k \in\{1 . . m\}$ and return the index of the machine with minimum load assuming $m>0$

```
fun min-arg \(::(\) nat \(\Rightarrow\) nat \() \Rightarrow\) nat \(\Rightarrow\) nat where
    \(\min -\arg T 0=1\)
\(\mid\) min-arg \(T(\) Suc \(x)=\)
    (let \(k=\) min-arg \(T x\)
        in if \(T(\) Suc \(x)<T k\) then (Suc \(x\) ) else \(k\) )
lemma min-correct:
    \(\forall x \in\{1 . . m\} . T(\) min-arg \(T m) \leq T x\)
    by (induction m) (auto simp: Let-def le-Suc-eq, force)
lemma min-in-range:
    \(k>0 \Longrightarrow(\) min-arg \(T k) \in\{1 . . k\}\)
    by (induction \(k\) ) (auto simp: Let-def, force + )
```

```
lemma add-job:
    assumes lb TAjx\in{1..m}
    shows lb (T (x:=T x + t (Suc j))) (A (x:=A x \cup{Suc j})) (Suc j)
        (is <lb ?T ?A ->)
proof -
    note lbrules = lbE[OF assms(1)]
    - Rule 1: A(x:=A x\cup{Suc j}) pairwise disjoint
    have NOTIN: }\foralli\in{1..m}.Suc j\not\inA i using lbrules(2) assms(2) by forc
    with lbrules(1) have }\foralli\in{1..m}. i\not=x\longrightarrowA i\cap(Ax\cup{Suc j})={
    using assms(2) by blast
    then have 1:}\forallx\in{1..m}.\forally\in{1..m}. x\not=y\longrightarrow?A x\cap?A y={
    using lbrules(1) by simp
- Rule 2: \(A(x:=A x \cup\{S u c j\})\) contains all jobs
    have}(\bigcupy\in{1..m}.?A y)=(\bigcupy\in{1..m}.A y)\cup{Suc j
    using UNION-fun-upd assms(2) by auto
    also have ... ={1..Suc j} unfolding lbrules(2) by auto
    finally have 2: (\bigcupy f {1..m}. ?A y)={1..Suc j} .
    _ Rule 3: A(x :=A x\cup{Suc j}) sums to T(x:=T x+t(Suc j))
    have (\sumi\in?A x.t i)=(\sumi\inAx\cup{Suc j}.t t ) by simp
    moreover have A x\cap{Suc j}={} using NOTIN assms(2) by blast
    moreover have finite (A x) finite {Suc j} using assms by simp+
    ultimately have (\sumi\in?A x. t i) = (\sumi\inA x.t i) +(\sumi\in{Suc j}.t i)
        using sum.union-disjoint by simp
    also have ... = Tx+t(Suc j) using lbrules(3) assms(2) by simp
    finally have (\sumi\in?A x.t i)=?T x by simp
    then have 3: }\foralli\in{1..m}.(\sumj\in?A i.tj)=?T
    using lbrules(3) assms(2) by simp
    from lbI[OF
qed
lemma makespan-mono:
    y \leqTx\Longrightarrow makespan (T (x:=y)) \leq makespan T
    Tx}\leqy\Longrightarrow\mathrm{ makespan T}\leq\mathrm{ makespan (T (x:=y))
    using f-Max0-mono by auto
lemma smaller-optimum:
assumes \(l b T A(S u c j)\)
shows \(\exists T^{\prime} A^{\prime}\). lb \(T^{\prime} A^{\prime} j \wedge\) makespan \(T^{\prime} \leq\) makespan \(T\)
proof -
note lbrules \(=l b E[O F\) assms \(]\)
have \(\exists x \in\{1\)..m\}. Suc \(j \in A x\) using lbrules(2) by auto
then obtain \(x\) where \(x\)-def: \(x \in\{1 . . m\}\) Suc \(j \in A x\)..
let ? \(T=T(x:=T x-t(\) Suc \(j))\)
let ? \(A=A(x:=A x-\{\) Suc \(j\})\)
```

— Rule 1: $A(x:=A x-\{S u c j\})$ pairwise disjoint
from lbrules (1) have $\forall i \in\{1 . . m\} . i \neq x \longrightarrow A i \cap(A x-\{S u c j\})=\{ \}$ using $x$-def(1) by blast
then have $1: \forall x \in\{1 . . m\} . \forall y \in\{1 . . m\} . x \neq y \longrightarrow ? A x \cap ? A y=\{ \}$
using lbrules(1) by auto

- Rule 2: $A(x:=A x-\{S u c j\})$ contains all jobs
have NOTIN: $\forall i \in\{1 . . m\} . i \neq x \longrightarrow S u c j \notin A i$ using lbrules(1) $x$-def by blast
then have $(\bigcup y \in\{1 . . m\} . ? A y)=(\bigcup y \in\{1 . . m\} . A y)-\{S u c j\}$
using UNION-fun-upd $x$-def by auto
also have $\ldots=\{1 . . j\}$ unfolding lbrules(2) by auto
finally have 2: $(\bigcup y \in\{1 . . m\}$. ? $A y)=\{1 . . j\}$.
— Rule 3: $A(x:=A x-\{$ Suc $j\})$ sums to $T(x:=T x-t(S u c j))$
have $\left(\sum i \in A x-\{S u c j\} . t i\right)=\left(\sum i \in A x . t i\right)-t(S u c j)$
by (simp add: sum-diff1-nat $x$-def(2))
also have $\ldots=T x-t$ (Suc j) using lbrules(3) $x$ - $\operatorname{def}(1)$ by $\operatorname{simp}$
finally have $\left(\sum i \in\right.$ ? $\left.A x . t i\right)=$ ? $T x$ by simp
then have 3: $\forall i \in\{1 . . m\} .\left(\sum j \in ? A\right.$ i. $\left.t j\right)=? T i$
using lbrules(3) $x$ - $\operatorname{def}(1)$ by $\operatorname{simp}$
- makespan is not larger
have $l b$ ? $T$ ? A $j \wedge$ makespan ? $T \leq$ makespan $T$
using $l b I\left[O F 1 \begin{array}{lll}1 & 2 & 3\end{array}\right]$ makespan-mono(1) by force
then show ?thesis by blast
qed
If the processing time $y$ does not contribute to the makespan, we can ignore it.

```
lemma remove-small-job:
    assumes makespan \((T(x:=T x+y)) \neq T x+y\)
    shows makespan \((T(x:=T x+y))=\) makespan \(T\)
proof -
    let ? \(T=T(x:=T x+y)\)
    have NOT-X: makespan ?T \(\neq ? T x\) using \(\operatorname{assms}(1)\) by \(\operatorname{simp}\)
    then have \(\exists i \in\{1 . . m\}\). makespan ? \(T=? T i \wedge i \neq x\)
        using makespan-correct(2) by metis
    then obtain \(i\) where \(i\)-def: \(i \in\{1 . . m\}\) makespan ? \(T=\) ? \(T i i \neq x\) by blast
    then have ?T \(i=T i\) using \(N O T-X\) by simp
    moreover from this have makespan \(T=T i\)
    by (metis \(i-\operatorname{def}(1,2)\) antisym-conv le-add1 makespan-mono(2) makespan-correct(1))
    ultimately show ?thesis using \(i-\operatorname{def}(2)\) by simp
qed
lemma greedy-makespan-no-jobs [simp]:
    makespan \((\lambda-.0)=0\)
    using \(m\)-gt-0 by (simp add: makespan-def')
```

```
lemma min-avg: \(m * T(\) min-arg \(T m) \leq\left(\sum i \in\{1 . . m\} . T i\right)\)
            (is \(\langle-* ? T \leq ? S\rangle)\)
proof -
    have \(\left(\sum-\in\{1 . . m\}\right.\). ? \(\left.T\right) \leq ? S\)
        using sum-mono[of \(\langle\{1 . . m\}\rangle\langle\lambda\)-. ? \(T\rangle T]\)
            and min-correct by blast
    then show? ?thesis by simp
qed
definition \(i n v_{1}::(\) nat \(\Rightarrow\) nat \() \Rightarrow(\) nat \(\Rightarrow\) nat set \() \Rightarrow\) nat \(\Rightarrow\) bool where
    \(\operatorname{inv}_{1} T A j=\left(l b T A j \wedge j \leq n \wedge\left(\forall T^{\prime} A^{\prime} . l b T^{\prime} A^{\prime} j \longrightarrow\right.\right.\) makespan \(T \leq 2 *\)
makespan \(\left.T^{\prime}\right)\) )
lemma \(i n v_{1} E\) :
    assumes \(i n v_{1} T A j\)
    shows \(l b T A j j \leq n\)
        \(l b T^{\prime} A^{\prime} j \Longrightarrow\) makespan \(T \leq 2 *\) makespan \(T^{\prime}\)
    using assms unfolding inv \(v_{1}\)-def by blast+
lemma \(i n v_{1} I\) :
    assumes \(l b T A j j \leq n \forall T^{\prime} A^{\prime}\). lb \(T^{\prime} A^{\prime} j \longrightarrow\) makespan \(T \leq 2 *\) makespan \(T^{\prime}\)
    shows \(i n v_{1} T A j\) using assms unfolding inv \(v_{1}\)-def by blast
lemma inv \(_{1}\)-step:
    assumes \(i n v_{1} T A j j<n\)
    shows \(\operatorname{inv}_{1}(T((\min -\arg T m):=T(\) min-arg \(T m)+t(S u c j)))\)
            \((A((\min -\arg T m):=A(\min -\arg T m) \cup\{S u c j\}))(S u c j)\)
        (is \(\left\langle i n v_{1} ? T\right.\) ? \(\left.A->\right)\)
proof -
    note invrules \(=\operatorname{inv}_{1} E[\) OF assms(1) \(]\)
    - Greedy is correct
    have \(L B\) : \(l b\) ?T ? A (Suc \(j\) )
        using add-job[OF invrules(1) min-in-range[OF m-gt-0]] by blast
    - Greedy maintains approximation factor
    have \(M K: \forall T^{\prime} A^{\prime}\). lb \(T^{\prime} A^{\prime}(S u c j) \longrightarrow\) makespan ? \(T \leq 2 *\) makespan \(T^{\prime}\)
    proof rule +
        fix \(T_{1} A_{1}\) assume \(l b T_{1} A_{1}(S u c j)\)
        from smaller-optimum [OF this]
        obtain \(T_{0} A_{0}\) where \(l b T_{0} A_{0} j\) makespan \(T_{0} \leq\) makespan \(T_{1}\) by blast
        then have \(I H\) : makespan \(T \leq 2 *\) makespan \(T_{1}\)
        using invrules(3) by force
    show makespan? \(T \leq 2 *\) makespan \(T_{1}\)
    proof (cases \(\langle m a k e s p a n ? T=T(\) min-arg \(T m)+t(S u c j)\rangle)\)
        case True
        have \(m * T\) (min-arg \(T m) \leq\left(\sum i \in\{1 . . m\}\right.\). \(T i\) ) by (rule min-avg)
        also have \(\ldots=\left(\sum i \in\{1 . . j\} . t i\right)\) by (rule lb-impl-job-sum[OF invrules(1)])
        finally have real \(m * T(\) min-arg \(T m) \leq\left(\sum i \in\{1 . . j\} . t i\right)\)
            by (auto dest: of-nat-mono)
```

```
            with \(m\)-gt- 0 have \(T(\) min-arg \(T m) \leq\left(\sum i \in\{1 . . j\} . t i\right) / m\)
            by (simp add: field-simps)
            then have \(T(\) min-arg \(T m) \leq\) makespan \(T_{1}\)
                using job-dist-lower-bound-makespan[OF \(\left.\left\langle l b T_{0} A_{0} j\right\rangle\right]\)
            and \(\left\langle m a k e s p a n ~ T_{0} \leq\right.\) makespan \(\left.T_{1}\right\rangle\) by linarith
            moreover have \(t(S u c j) \leq\) makespan \(T_{1}\)
            using job-lower-bound-makespan[OF \(\left\langle l b T_{1} A_{1}\right.\) (Suc j) 〉] by simp
            ultimately show ?thesis unfolding True by simp
        next
            case False show ?thesis using remove-small-job[OF False] IH by simp
        qed
    qed
    from \(i n v_{1} I[O F L B-M K]\) show ?thesis using assms(2) by simp
qed
lemma simple-greedy-approximation:
VARS TA ij
\{True \(\}\)
\(T:=(\lambda-.0)\);
\(A:=(\lambda-.\{ \}) ;\)
\(j:=0\);
WHILE \(j<n\) INV \(\left\{i n v_{1} T A j\right\} D O\)
    \(i:=\) min-arg \(T\) m;
    \(j:=\) (Suc j);
    \(A:=A(i:=A(i) \cup\{j\}) ;\)
    \(T:=T(i:=T(i)+t j)\)
OD
\(\left\{l b T A n \wedge\left(\forall T^{\prime} A^{\prime}\right.\right.\). lb \(T^{\prime} A^{\prime} n \longrightarrow\) makespan \(T \leq 2 *\) makespan \(\left.\left.T^{\prime}\right)\right\}\)
proof (vcg, goal-cases)
    case ( \(1 T A i j\) )
    then show ?case by (simp add: lb-def inv \(v_{1}\)-def)
next
    case (2 TAij)
    then show? case using inv \(v_{1}\)-step by simp
next
    case (3 TAij)
    then show ?case unfolding inv \(v_{1}\) def by force
qed
definition sorted :: nat \(\Rightarrow\) bool where
    sorted \(j=(\forall x \in\{1 . . j\} . \forall y \in\{1 . . x\} . t x \leq t y)\)
lemma sorted-smaller \([\) simp \(]: \llbracket\) sorted \(j ; j \geq j^{\prime} \rrbracket \Longrightarrow\) sorted \(j^{\prime}\)
    unfolding sorted-def by simp
lemma j-gt-m-pigeonhole:
    assumes \(l b T A j j>m\)
    shows \(\exists x \in\{1 . . j\} . \exists y \in\{1 . . j\} . \exists z \in\{1 . . m\} . x \neq y \wedge x \in A z \wedge y \in A z\)
proof -
```

```
    have }\forallx\in{1..j}. \existsy\in{1..m}. x\inA
    using lbE(2)[OF assms(1)] by blast
    then have }\existsf.\forallx\in{1..j}. x\inA(fx)\wedgefx\in{1..m} by meti
    then obtain f}\mathrm{ where f-def: }\forallx\in{1..j}.x\inA(fx)\wedgefx\in{1..m} ..
    then have card (f'{1..j})\leq\operatorname{card {1..m}}
    by (meson card-mono finite-atLeastAtMost image-subset-iff)
    also have ... < card {1..j} using assms(2) by simp
    finally have card (f'{1..j})<card {1..j} .
    then have }\neg\operatorname{inj-on f {1..j} using pigeonhole by blast
    then have \existsx\in{1..j}. \existsy\in{1..j}. x = y^fx=fy
    unfolding inj-on-def by blast
    then show ?thesis using f-def by metis
qed
```

If $T$ and $A$ are a correct load balancing for $j$ jobs and $m$ machines with $j$ $>m$, and the jobs are sorted in descending order, then there exists a machine $x \in\{1 . . m\}$ whose load is at least twice as large as the processing time of job $j$.
lemma sorted-job-lower-bound-machine:
assumes $l b T A j j>m$ sorted $j$
shows $\exists x \in\{1$.. $m\}$. $2 * t j \leq T x$
proof -

- Step 1: Obtaining the jobs
note lbrules $=l b E[O F \operatorname{assms}(1)]$
obtain $j_{1} j_{2} x$ where $*$ :
$j_{1} \in\{1 . . j\} j_{2} \in\{1 . . j\} x \in\{1 . . m\} j_{1} \neq j_{2} j_{1} \in A x j_{2} \in A x$
using $j$-gt-m-pigeonhole[OF assms(1,2)] by blast
- Step 2: Jobs contained in sum
have finite ( $A x$ ) using assms(1) *(3) by simp
then have SUM: $\left(\sum i \in A x . t i\right)=t j_{1}+t j_{2}+\left(\sum i \in A x-\left\{j_{1}\right\}-\left\{j_{2}\right\} . t\right.$
i)
using $*(4-6)$ by (simp add: sum.remove)
- Step 3: Proof of lower bound
have $t j \leq t j_{1} t j \leq t j_{2}$
using $\operatorname{assms}(3) *(1-2)$ unfolding sorted-def by auto
then have $2 * t j \leq t j_{1}+t j_{2}$ by simp
also have $\ldots \leq\left(\sum i \in A x\right.$. $\left.t i\right)$ unfolding $S U M$ by simp
finally have 2 $* t j \leq T x$ using $\operatorname{lbrules}(3) *(3)$ by $\operatorname{simp}$
then show ?thesis using $*(3)$ by blast
qed
Reasoning analogous to job-lower-bound-makespan.
lemma sorted-job-lower-bound-makespan:
assumes $l b T A j j>m$ sorted $j$
shows $2 * t j \leq$ makespan $T$
proof -
obtain $x$ where $x$-def: $x \in\{1 . . m\} 2 * t j \leq T x$
using sorted-job-lower-bound-machine[OF assms] ..
with makespan-correct(1) have $T x \leq$ makespan $T$ by blast with $x$ - def(2) show ?thesis by simp
qed
lemma min-zero:
assumes $x \in\{1 . . k\} T x=0$
shows $T(\min -\arg T k)=0$
using assms(1)
proof (induction $k$ )
case (Suc k)
show ? case proof (cases $\langle x=$ Suc $k\rangle$ )
case True
then show ?thesis using assms(2) by (simp add: Let-def)
next
case False
with $S u c$ have $T(\min -\arg T k)=0$ by $\operatorname{simp}$
then show ?thesis by simp
qed
qed $\operatorname{simp}$
lemma min-zero-index:
assumes $x \in\{1 . . k\} T x=0$
shows min-arg $T k \leq x$
using assms(1)
proof (induction $k$ )
case (Suc k)
show ?case proof (cases $\langle x=$ Suc $k\rangle$ )
case True
then show ?thesis using min-in-range $[$ of Suc $k$ ] by simp
next
case False
with Suc.prems have $x \in\{1 . . k\}$ by simp
from min-zero[OF this, of T] assms(2) Suc.IH[OF this]
show? ?thesis by simp
qed
qed $\operatorname{simp}$
definition $\mathrm{inv}_{2}::($ nat $\Rightarrow$ nat $) \Rightarrow($ nat $\Rightarrow$ nat set $) \Rightarrow$ nat $\Rightarrow$ bool where
$i n v_{2} T A j=(l b T A j \wedge j \leq n$
$\wedge\left(\forall T^{\prime} A^{\prime} . l b T^{\prime} A^{\prime} j \longrightarrow\right.$ makespan $T \leq 3 / 2 *$ makespan $\left.T^{\prime}\right)$
$\wedge(\forall x>j . T x=0)$
$\wedge\left(j \leq m \longrightarrow\right.$ makespan $\left.\left.T=\operatorname{Max}_{0}\left(t^{\prime}\{1 . . j\}\right)\right)\right)$
lemma $i n v_{2} E$ :
assumes $\operatorname{inv}_{2} T A j$
shows $l b T A j j \leq n$
$l b T^{\prime} A^{\prime} j \Longrightarrow$ makespan $T \leq 3 / 2 *$ makespan $T^{\prime}$
$\forall x>j . T x=0 j \leq m \Longrightarrow$ makespan $T=\operatorname{Max}_{0}\left(t^{\prime}\{1 . . j\}\right)$
using assms unfolding $i n v_{2}$-def by blast+


## lemma $i n v_{2} I$ :

assumes $l b T A j j \leq n$

$$
\begin{aligned}
& \forall T^{\prime} A^{\prime} . l b T^{\prime} A^{\prime} j \longrightarrow \text { makespan } T \leq 3 / 2 * \text { makespan } T^{\prime} \\
& \forall x>j . T x=0 \\
& j \leq m \Longrightarrow \text { makespan } T=\operatorname{Max}_{0}(t ‘\{1 . . j\})
\end{aligned}
$$

shows $i n v_{2} T A j$
unfolding $i n v_{2}$-def using assms by blast
lemma $i n v_{2}$-step:
assumes sorted $n i n v_{2} T A j j<n$
shows $\operatorname{inv}_{2}(T($ min-arg $T m:=T($ min-arg $T m)+t(S u c j)))$
$(A(\min -\arg T m:=A(\min -\arg T m) \cup\{S u c j\}))(S u c j)$
(is $\left\langle i n v_{2}\right.$ ? $T$ ? $A$->)
proof (cases $\langle S u c j>m\rangle$ )
case True note invrules $=i n v_{2} E[$ OF assms(2)]

- Greedy is correct
have $L B$ : lb ? $T$ ? A (Suc $j$ )
using add-job[OF invrules(1) min-in-range[OF m-gt-0]] by blast
- Greedy maintains approximation factor
have $M K: \forall T^{\prime} A^{\prime}$.lb $T^{\prime} A^{\prime}(S u c j) \longrightarrow$ makespan ? $T \leq 3 / 2 *$ makespan $T^{\prime}$
proof rule+
fix $T_{1} A_{1}$ assume $l b T_{1} A_{1}$ (Suc $j$ )
from smaller-optimum[OF this]
obtain $T_{0} A_{0}$ where $l b T_{0} A_{0} j$ makespan $T_{0} \leq$ makespan $T_{1}$ by blast
then have $I H:$ makespan $T \leq 3 / 2 *$ makespan $T_{1}$
using invrules(3) by force
show makespan? $T \leq 3 / 2 *$ makespan $T_{1}$
proof (cases 〈makespan ? $T=T($ min-arg $T m)+t(S u c j)\rangle)$
case True
have $m * T$ (min-arg $T m) \leq\left(\sum i \in\{1 . . m\}\right.$. $T i$ ) by (rule min-avg)
also have $\ldots=\left(\sum i \in\{1 . . j\} . t i\right)$ by (rule lb-impl-job-sum $[\operatorname{OF}$ invrules $\left.(1)]\right)$
finally have real $m * T($ min-arg $T m) \leq\left(\sum i \in\{1 . . j\} . t i\right)$
by (auto dest: of-nat-mono)
with $m$-gt- 0 have $T($ min-arg $T m) \leq\left(\sum i \in\{1 . . j\} . t i\right) / m$ by (simp add: field-simps)
then have $T($ min-arg $T m) \leq$ makespan $T_{1}$
using job-dist-lower-bound-makespan $\left[O F\left\langle l b T_{0} A_{0} j\right\rangle\right]$
and $\left\langle\right.$ makespan $T_{0} \leq$ makespan $\left.T_{1}\right\rangle$ by linarith
moreover have $2 * t(S u c j) \leq$ makespan $T_{1}$
using sorted-job-lower-bound-makespan[OF $\left.\left\langle l b T_{1} A_{1}(S u c j)\right\rangle\langle S u c j>m\rangle\right]$
and $\operatorname{assms}(1,3)$ by $\operatorname{simp}$
ultimately show ?thesis unfolding True by simp


## next

case False show ?thesis using remove-small-job[OF False] IH by simp qed
qed
have $\forall x>$ Suc $j$. ? $T x=0$
using invrules(4) min-in-range[OF m-gt-0, of T] True by simp with inv $_{2} I[O F L B-M K]$ show ?thesis using $\operatorname{assms}(3)$ True by simp next
case False
then have $I N-R A N G E: S u c j \in\{1 . . m\}$ by simp
note invrules $=i n v_{2} E[O F$ assms(2)]
then have $T(S u c j)=0$ by blast

- Greedy is correct
have $L B$ : $l b$ ?T ? A (Suc $j$ )
using add-job[OF invrules(1) min-in-range[OF m-gt-0]] by blast
- Greedy is trivially optimal
from $I N$-RANGE $\langle T(S u c j)=0\rangle$ have min-arg $T m \leq S u c j$
using min-zero-index by blast
with invrules(4) have EMPTY: $\forall x>$ Suc $j$. ?T $x=0$ by simp
from $I N-R A N G E<T(S u c j)=0\rangle$ have $T($ min-arg $T m)=0$
using min-zero by blast
with fun-upd-f-Max ${ }_{0}[O F$ min-in-range $[O F ~ m-g t-0]]$ invrules(5) False
have TRIV: makespan? $T=\operatorname{Max}_{0}\left(t^{\prime}\{1 . . S u c j\}\right)$ unfolding $f$-Max $x_{0}$-equiv $[$ symmetric $]$
by $\operatorname{simp}$
have $M K$ : $\forall T^{\prime} A^{\prime}$. lb $T^{\prime} A^{\prime}(S u c j) \longrightarrow$ makespan ? $T \leq 3 / 2 *$ makespan $T^{\prime}$
by (auto simp: TRIV[folded $f$-Max $0_{0}$-equiv] dest!: max-job-lower-bound-makespan[folded $f$-Max $0_{0}$-equiv] $)$
from $i n v_{2} I[$ OF LB - MK EMPTY TRIV] show ?thesis using assms(3) by simp qed
lemma sorted-greedy-approximation:
sorted $n \Longrightarrow$ VARS TAij
\{True $\}$
$T:=(\lambda-.0)$;
$A:=(\lambda-.\{ \}) ;$
$j:=0$;
WHILE $j<n$ INV $\left\{i n v_{2} T A j\right\} D O$
$i:=$ min-arg $T$ m;
$j:=$ (Suc $j$ );
$A:=A(i:=A(i) \cup\{j\}) ;$
$T:=T(i:=T(i)+t j)$
$O D$
$\left\{l b T A n \wedge\left(\forall T^{\prime} A^{\prime} . l b T^{\prime} A^{\prime} n \longrightarrow\right.\right.$ makespan $T \leq 3 / 2 *$ makespan $\left.\left.T^{\prime}\right)\right\}$
proof (vcg, goal-cases)
case (1 T A i j)
then show ?case by (simp add: lb-def inv $v_{2}$-def)
next
case (2 $T A i j$ )
then show ?case using inv $\mathrm{in}_{2}$-step by simp
next
case (3 TAij)

```
    then show ?case unfolding inv2-def by force
qed
end
```

end

## 5 Bin Packing

```
theory Approx-BP-Hoare
    imports Complex-Main HOL-Hoare.Hoare-Logic HOL-Library.Disjoint-Sets
begin
```

The algorithm and proofs are based on the work by Berghammer and Reuter [2].

### 5.1 Formalization of a Correct Bin Packing

Definition of the unary operator $\llbracket \rrbracket \rrbracket$ from the article. $B$ will only be wrapped into a set if it is non-empty.

```
definition wrap :: 'a set \(\Rightarrow\) 'a set set where
    wrap \(B=(\) if \(B=\{ \}\) then \(\{ \}\) else \(\{B\}\) )
lemma wrap-card:
    card \((\) wrap \(B) \leq 1\)
    unfolding wrap-def by auto
```

If $M$ and $N$ are pairwise disjoint with $V$ and not yet contained in V , then the union of $M$ and $N$ is also pairwise disjoint with $V$.

```
lemma pairwise-disjnt-Un:
    assumes pairwise disjnt ({M}\cup{N}\cupV)M\not\inVN\not\inV
    shows pairwise disjnt ({M\cupN}\cupV)
    using assms unfolding pairwise-def by auto
```

        A Bin Packing Problem is defined like in the article:
    locale BinPacking =
fixes $U$ :: ' $a$ set - A finite, non-empty set of objects
and $w::{ }^{\prime} a \Rightarrow$ real - A mapping from objects to their respective weights
(positive real numbers)
and $c::$ nat - The maximum capacity of a bin (a natural number)
and $S::$ ' $a$ set - The set of small objects (weight no larger than $1 / 2$ of $c$ )
and $L::$ 'a set - The set of large objects (weight larger than $1 / 2$ of $c$ )
assumes weight: $\forall u \in U .0<w(u) \wedge w(u) \leq c$
and $U$-Finite: finite $U$
and $U-N E: U \neq\{ \}$
and $S$-def: $S=\{u \in U . w(u) \leq c / 2\}$
and $L$-def: $L=U-S$
begin

In the article, this is defined as $w$ as well. However, to avoid ambiguity, we will abbreviate the weight of a bin as $W$.
abbreviation $W$ :: 'a set $\Rightarrow$ real where
$W B \equiv\left(\sum u \in B . w(u)\right)$
$P$ constitutes as a correct bin packing if $P$ is a partition of $U$ (as defined in partition-on-def) and the weights of the bins do not exceed their maximum capacity $c$.
definition $b p::$ ' $a$ set set $\Rightarrow$ bool where bp $P \longleftrightarrow$ partition-on $U P \wedge(\forall B \in P . W(B) \leq c)$

## lemma $b p E$ :

assumes $b p P$
shows pairwise disjnt $P\} \notin P \bigcup P=U \forall B \in P . W(B) \leq c$
using assms unfolding bp-def partition-on-def by blast+

## lemma $b p I$ :

assumes pairwise disjnt $P\} \notin P \bigcup P=U \forall B \in P . W(B) \leq c$
shows bp $P$
using assms unfolding bp-def partition-on-def by blast
Although we assume the $S$ and $L$ sets as given, manually obtaining them from $U$ is trivial and can be achieved in linear time. Proposed by the article [2].

```
lemma \(S\)-L-set-generation:
VARS SL Wu
    \{True \(\}\)
    \(S:=\{ \} ; L:=\{ \} ; W:=U\);
    WHILE \(W \neq\{ \}\)
    \(I N V\{W \subseteq U \wedge S=\{v \in U-W . w(v) \leq c / 2\} \wedge L=\{v \in U-W \cdot w(v)\)
> c/2\}\} DO
    \(u:=(S O M E \quad u . u \in W)\);
    IF 2 * \(w(u) \leq c\)
    THEN \(S:=S \cup\{u\}\)
    ELSE \(L:=L \cup\{u\} F I\);
    \(W:=W-\{u\}\)
    OD
    \(\{S=\{v \in U \cdot w(v) \leq c /\) 2 \(\} \wedge L=\{v \in U \cdot w(v)>c / 2\}\}\)
    by vcg (auto simp: some-in-eq)
```


### 5.2 The Proposed Approximation Algorithm

### 5.2.1 Functional Correctness

According to the article, $i n v_{1}$ holds if $P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup$ $\{\{v\} \mid v . v \in V\}$ is a correct solution for the bin packing problem [2]. However, various assumptions made in the article seem to suggest that more information is demanded from this invariant and, indeed, mere correctness
(as defined in $b p$-def) does not appear to suffice. To amend this, four additional conjuncts have been added to this invariant, whose necessity will be explained in the following proofs. It should be noted that there may be other (shorter) ways to amend this invariant. This approach, however, makes for rather straight-forward proofs, as these conjuncts can be utilized and proved in relatively few steps.
definition inv $_{1}::$ 'a set set $\Rightarrow$ 'a set set $\Rightarrow$ 'a set $\Rightarrow$ 'a set $\Rightarrow$ ' $a$ set $\Rightarrow$ bool where $\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2} V \longleftrightarrow b p\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in$ $V\})$ - A correct solution to the bin packing problem

$$
\wedge \bigcup\left(P_{1} \cup \text { wrap } B_{1} \cup P_{2} \cup \text { wrap } B_{2}\right)=U-V-\text { The partial }
$$ solution does not contain objects that have not yet been assigned

$$
\wedge B_{1} \notin\left(P_{1} \cup P_{2} \cup w r a p B_{2}\right)-B_{1} \text { is distinct from all the other }
$$

bins
$\wedge B_{2} \notin\left(P_{1} \cup\right.$ wrap $\left.B_{1} \cup P_{2}\right)-B_{2}$ is distinct from all the other
bins
$\wedge\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right) \cap\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)=\{ \}$ - The first and second partial solutions are disjoint from each other.

```
lemma \(i n v_{1} E\) :
    assumes \(i n v_{1} P_{1} P_{2} B_{1} B_{2} V\)
    shows \(b p\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2} \cup\{\{v\} \mid v . v \in V\}\right)\)
        and \(\cup\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2}\right)=U-V\)
        and \(B_{1} \notin\left(P_{1} \cup P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\)
        and \(B_{2} \notin\left(P_{1} \cup\right.\) wrap \(\left.B_{1} \cup P_{2}\right)\)
        and \(\left(P_{1} \cup\right.\) wrap \(\left.B_{1}\right) \cap\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)=\{ \}\)
    using assms unfolding inv \(v_{1}\) def by auto
lemma \(i n v_{1} I\) :
    assumes \(b p\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2} \cup\{\{v\} \mid v . v \in V\}\right)\)
    and \(\cup\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2}\right)=U-V\)
    and \(B_{1} \notin\left(P_{1} \cup P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\)
    and \(B_{2} \notin\left(P_{1} \cup\right.\) wrap \(\left.B_{1} \cup P_{2}\right)\)
    and \(\left(P_{1} \cup\right.\) wrap \(\left.B_{1}\right) \cap\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)=\{ \}\)
    shows \(\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2} V\)
    using assms unfolding \(i n v_{1}\)-def by blast
```

lemma wrap-Un [simp]: wrap $(M \cup\{x\})=\{M \cup\{x\}\}$ unfolding wrap-def by simp
lemma wrap-empty [simp]: wrap $\}=\{ \}$ unfolding wrap-def by simp
lemma wrap-not-empty $[$ simp $]: M \neq\{ \} \longleftrightarrow$ wrap $M=\{M\}$ unfolding wrap-def by $\operatorname{simp}$

If $i n v_{1}$ holds for the current partial solution, and the weight of an object $u \in V$ added to $B_{1}$ does not exceed its capacity, then $i n v_{1}$ also holds if $B_{1}$ and $\{u\}$ are replaced by $B_{1} \cup\{u\}$.
lemma inv ${ }_{1}$-step $A$ :
assumes $\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2} V u \in V W\left(B_{1}\right)+w(u) \leq c$
shows $\operatorname{inv}_{1} P_{1} P_{2}\left(B_{1} \cup\{u\}\right) B_{2}(V-\{u\})$

```
proof -
    note invrules = inv 
```

In the proof for Theorem 3.2 of the article it is erroneously argued that if $P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup \operatorname{wrap} B_{2} \cup\{\{v\} \mid v . v \in V\}$ is a partition of $U$, then the same holds if $B_{1}$ is replaced by $B_{1} \cup\{u\}$. This is, however, not necessarily the case if $B_{1}$ or $\{u\}$ are already contained in the partial solution. Suppose $P_{1}$ contains the non-empty bin $B_{1}$, then $P_{1} \cup$ wrap $B_{1}$ would still be pairwise disjoint, provided $P_{1}$ was pairwise disjoint before, as the union simply ignores the duplicate $B_{1}$. Now, if the algorithm modifies $B_{1}$ by adding an element from $V$ such that $B_{1}$ becomes some non-empty $B_{1}^{\prime}$ with $B_{1} \cap B_{1}^{\prime} \neq \emptyset$ and $B_{1}^{\prime} \notin P_{1}$, one can see that this property would no longer be preserved. To avoid such a situation, we will use the first additional conjunct in $i n v_{1}$ to ensure that $\{u\}$ is not yet contained in the partial solution, and the second additional conjunct to ensure that $B_{1}$ is not yet contained in the partial solution.
have NOTIN: $\forall M \in P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$. $u \notin M$
using invrules(2) assms(2) by blast
have $\{\{v\} \mid v . v \in V\}=\{\{u\}\} \cup\{\{v\} \mid v . v \in V-\{u\}\}$ using assms(2) by blast
then have pairwise disjnt $\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup(\{\{u\}\} \cup\{\{v\} \mid v$. $v \in V-\{u\}\}))$
using bprules(1) assms(2) by simp
then have pairwise disjnt (wrap $B_{1} \cup\{\{u\}\} \cup P_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v$. $v \in V-\{u\}\}$ ) by (simp add: Un-commute)
then have assm: pairwise disjnt (wrap $B_{1} \cup\{\{u\}\} \cup\left(P_{1} \cup P_{2} \cup\right.$ wrap $B_{2} \cup$ $\{\{v\} \mid v . v \in V-\{u\}\})$ ) by (simp add: Un-assoc)
have pairwise disjnt $\left(\left\{B_{1} \cup\{u\}\right\} \cup\left(P_{1} \cup P_{2} \cup\right.\right.$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-$ $\{u\}\}$ ))
proof (cases $\left.\left\langle B_{1}=\{ \}\right\rangle\right)$
case True with assm show ?thesis by simp
next
case False
with assm have assm: pairwise disjnt $\left(\left\{B_{1}\right\} \cup\{\{u\}\} \cup\left(P_{1} \cup P_{2} \cup\right.\right.$ wrap $B_{2}$ $\cup\{\{v\} \mid v . v \in V-\{u\}\}))$ by simp
from NOTIN have $\{u\} \notin P_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$ by blast
from pairwise-disjnt-Un[OF assm - this] invrules $(2,3)$ show ?thesis using False by auto
qed
then have 1: pairwise disjnt $\left(P_{1} \cup\right.$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\}$ $\mid v . v \in V-\{u\}\})$
unfolding wrap-Un by simp

- Rule 2: No empty sets
from bprules(2) have 2: $\left\} \notin P_{1} \cup\right.$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\}$ $\mid v . v \in V-\{u\}\}$
unfolding wrap-def by simp
- Rule 3: Union preserved
from $\operatorname{bprules}(3)$ have $\bigcup\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{u\}\} \cup\{\{v\} \mid v . v$ $\in V-\{u\}\})=U$
using assms(2) by blast
then have 3: $\cup\left(P_{1} \cup\right.$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-$ $\{u\}\})=U$
unfolding wrap-def by force
- Rule 4: Weights below capacity
have $0<w u$ using weight assms(2) bprules(3) by blast
have finite $B_{1}$ using bprules(3) U-Finite by (cases $\left\langle B_{1}=\{ \}\right\rangle$ ) auto
then have $W\left(B_{1} \cup\{u\}\right) \leq W B_{1}+w u$ using $\langle 0<w u\rangle$ by (cases $\left.\left\langle u \in B_{1}\right\rangle\right)$
(auto simp: insert-absorb)
also have $\ldots \leq c$ using $\operatorname{assms}(3)$.
finally have $W\left(B_{1} \cup\{u\}\right) \leq c$.
then have $\forall B \in \operatorname{wrap}\left(B_{1} \cup\{u\}\right)$. W $B \leq c$ unfolding wrap-Un by blast
moreover have $\forall B \in P_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$. W $B \leq c$
using bprules(4) by blast
ultimately have $4: \forall B \in P_{1} \cup \operatorname{wrap}\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v$ $\in V-\{u\}\}$. W $B \leq c$ by blast
from $b p I[O F 1234]$ have $1: b p\left(P_{1} \cup\right.$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $B_{2} \cup$ $\{\{v\} \mid v . v \in V-\{u\}\})$.
- Auxiliary information is preserved
have $u \in U$ using assms(2) bprules(3) by blast
then have $R$ : $U-(V-\{u\})=U-V \cup\{u\}$ by blast
have $L: \cup\left(P_{1} \cup\right.$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $\left.B_{2}\right)=\bigcup\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2}$ $\cup$ wrap $\left.B_{2}\right) \cup\{u\}$
unfolding wrap-def using NOTIN by auto
have 2: $\cup\left(P_{1} \cup\right.$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $\left.B_{2}\right)=U-(V-\{u\})$
unfolding $L R$ invrules(2) ..
have 3: $B_{1} \cup\{u\} \notin P_{1} \cup P_{2} \cup$ wrap $B_{2}$
using NOTIN by auto
have 4: $B_{2} \notin P_{1} \cup$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2}$
using invrules(4) NOTIN unfolding wrap-def by fastforce
have 5: $\left(P_{1} \cup\right.$ wrap $\left.\left(B_{1} \cup\{u\}\right)\right) \cap\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)=\{ \}$
using invrules(5) NOTIN unfolding wrap-Un by auto
from inv $_{1} I[O F 12345]$ show ?thesis.
qed
If $i n v_{1}$ holds for the current partial solution, and the weight of an object $u \in V$ added to $B_{2}$ does not exceed its capacity, then $i n v_{1}$ also holds if $B_{2}$ and $\{u\}$ are replaced by $B_{2} \cup\{u\}$.
lemma inv $v_{1}-$ step $B$ :
assumes inv $P_{1} P_{2} B_{1} B_{2} V u \in V W B_{2}+w u \leq c$
shows $\operatorname{inv}_{1}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right) P_{2}\{ \}\left(B_{2} \cup\{u\}\right)(V-\{u\})$

```
proof -
    note invrules = inv }E[OF assms(1)] and bprules =bpE[OF invrules(1)]
```

The argumentation here is similar to the one in $i n v_{1}-\operatorname{step} A$ with $B_{1}$ replaced with $B_{2}$ and using the first and third additional conjuncts of $i n v_{1}$ to amend the issue, instead of the first and second.

```
    have NOTIN: \(\forall M \in P_{1} \cup\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(B_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}\).
```

$u \notin M$
using invrules(2) assms(2) by blast
have $\{\{v\} \mid v . v \in V\}=\{\{u\}\} \cup\{\{v\} \mid v . v \in V-\{u\}\}$
using assms(2) by blast
then have pairwise disjnt $\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{u\}\} \cup\{\{v\} \mid v$.
$v \in V-\{u\}\})$
using bprules(1) assms(2) by simp
then have assm: pairwise disjnt (wrap $B_{2} \cup\{\{u\}\} \cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$
$\{\{v\} \mid v . v \in V-\{u\}\}))$
by (simp add: Un-assoc Un-commute)
have pairwise disjnt $\left(\left\{B_{2} \cup\{u\}\right\} \cup\left(P_{1} \cup\right.\right.$ wrap $B_{1} \cup P_{2} \cup\{\{v\} \mid v . v \in V-$
$\{u\}\})$ )
proof (cases $\left\langle B_{2}=\{ \}\right\rangle$ )
case True with assm show ?thesis by simp
next
case False
with assm have assm: pairwise disjnt $\left(\left\{B_{2}\right\} \cup\{\{u\}\} \cup\left(P_{1} \cup\right.\right.$ wrap $B_{1} \cup P_{2}$
$\cup\{\{v\} \mid v . v \in V-\{u\}\}))$ by simp
from NOTIN have $\{u\} \notin P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$ by
blast
from pairwise-disjnt-Un [OF assm - this] invrules(2,4) show ?thesis
using False by auto
qed
then have 1: pairwise disjnt $\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2} \cup\right.$ wrap $\left(B_{2} \cup\{u\}\right)$
$\cup\{\{v\} \mid v . v \in V-\{u\}\})$
unfolding wrap-Un by simp
- Rule 2: No empty sets
from $\operatorname{bprules}(2)$ have 2: $\left\} \notin P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2} \cup\right.$ wrap $\left(B_{2} \cup\right.$
$\{u\}) \cup\{\{v\} \mid v . v \in V-\{u\}\}$
unfolding wrap-def by simp

- Rule 3: Union preserved
from $\operatorname{bprules}(3)$ have $\bigcup\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{u\}\} \cup\{\{v\} \mid v . v$ $\in V-\{u\}\})=U$
using assms(2) by blast
then have 3: $\bigcup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2} \cup \operatorname{wrap}\left(B_{2} \cup\{u\}\right) \cup\{\{v\} \mid v\right.$. $v \in V-\{u\}\})=U$
unfolding wrap-def by force
- Rule 4: Weights below capacity
have $0<w u$ using weight assms(2) bprules(3) by blast
have finite $B_{2}$ using bprules(3) U-Finite by (cases $\left\langle B_{2}=\{ \}\right\rangle$ ) auto
then have $W\left(B_{2} \cup\{u\}\right) \leq W B_{2}+w u$ using $\langle 0<w u\rangle$ by (cases $\left\langle u \in B_{2}\right\rangle$ )
(auto simp: insert-absorb)
also have $\ldots \leq c$ using $\operatorname{assms}(3)$.
finally have $W\left(B_{2} \cup\{u\}\right) \leq c$.
then have $\forall B \in \operatorname{wrap}\left(B_{2} \cup\{u\}\right)$. W $B \leq c$ unfolding wrap-Un by blast
moreover have $\forall B \in P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$. W $B \leq c$ using bprules(4) by blast
ultimately have $4: \forall B \in P_{1} \cup$ wrap $B_{1} \cup \operatorname{wrap}\{ \} \cup P_{2} \cup$ wrap $\left(B_{2} \cup\{u\}\right) \cup$ $\{\{v\} \mid v . v \in V-\{u\}\} . W B \leq c$
by auto
from $b p I[O F 1234]$ have $1: b p\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\{ \} \cup P_{2} \cup$ wrap $\left(B_{2}\right.$ $\cup\{u\}) \cup\{\{v\} \mid v . v \in V-\{u\}\})$.
- Auxiliary information is preserved
have $u \in U$ using assms(2) bprules(3) by blast
then have $R$ : $U-(V-\{u\})=U-V \cup\{u\}$ by blast
have $L: \cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\{ \} \cup P_{2} \cup$ wrap $\left.\left(B_{2} \cup\{u\}\right)\right)=\bigcup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2} \cup\right.$ wrap $\left.B_{2}\right) \cup\{u\}$
unfolding wrap-def using NOTIN by auto
have 2: $\cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2} \cup\right.$ wrap $\left.\left(B_{2} \cup\{u\}\right)\right)=U-(V-$ $\{u\}$ )
unfolding $L R$ using invrules(2) by simp
have 3: $\left\} \notin P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $\left(B_{2} \cup\{u\}\right)$
using $b p E(2)[O F 1]$ by $\operatorname{simp}$
have 4: $B_{2} \cup\{u\} \notin P_{1} \cup$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2}\right.$ using NOTIN by auto
have 5: $\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\}) \cap\left(P_{2} \cup \operatorname{wrap}\left(B_{2} \cup\{u\}\right)\right)=\{ \}$
using invrules(5) NOTIN unfolding wrap-empty wrap-Un by auto
from $\operatorname{inv}_{1} I[$ OF 1234 5] show ?thesis.
qed
If $i n v_{1}$ holds for the current partial solution, then $i n v_{1}$ also holds if $B_{1}$ and $B_{2}$ are added to $P_{1}$ and $P_{2}$ respectively, $B_{1}$ is emptied and $B_{2}$ initialized with $u \in V$.
lemma inv $_{1}$-step $C$ :
assumes $\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2} V u \in V$
shows $\operatorname{inv}_{1}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)\{ \}\{u\}(V-\{u\})$
proof -
note invrules $=i n v_{1} E[O F \operatorname{assms}(1)]$
- Rule 1-4: Correct Bin Packing
have $P_{1} \cup$ wrap $B_{1} \cup$ wrap $\left\} \cup\left(P_{2} \cup\right.\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\{u\} \cup\{\{v\} \mid v . v \in V$
$-\{u\}\}$
$=P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{u\}\} \cup\{\{v\} \mid v . v \in V-\{u\}\}$
by (metis (no-types, lifting) Un-assoc Un-empty-right insert-not-empty wrap-empty wrap-not-empty)
also have $\ldots=P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V\}$
using assms(2) by auto
finally have $E Q: P_{1} \cup$ wrap $B_{1} \cup$ wrap $\left\} \cup\left(P_{2} \cup\right.\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\{u\} \cup$ $\{\{v\} \mid v . v \in V-\{u\}\}$

$$
=P_{1} \cup \text { wrap } B_{1} \cup P_{2} \cup \text { wrap } B_{2} \cup\{\{v\} \mid v . v \in V\} .
$$

from invrules $(1)$ have $1: b p\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\{ \} \cup\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\{u\} \cup\{\{v\} \mid v . v \in V-\{u\}\})$
unfolding $E Q$.

- Auxiliary information is preserved
have NOTIN: $\forall M \in P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$. $u \notin M$
using invrules(2) assms(2) by blast
have $u \in U$ using assms(2) bpE(3)[OF invrules(1)] by blast
then have $R: U-(V-\{u\})=U-V \cup\{u\}$ by blast
have $L: \cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\{ \} \cup\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\left.\{u\}\right)=\bigcup\left(P_{1}\right.$ $\cup$ wrap $B_{1} \cup$ wrap $\left\} \cup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \cup\{u\}$ unfolding wrap-def using NOTIN by auto
have 2: $\cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup\left(P_{2} \cup\right.\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\left.\{u\}\right)=U-(V$ $-\{u\})$
unfolding $L R$ using invrules(2) by auto
have 3: $\left\} \notin P_{1} \cup\right.$ wrap $B_{1} \cup\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\{u\}$ using $b p E$ (2)[OF 1] by simp
have 4: $\{u\} \notin P_{1} \cup$ wrap $B_{1} \cup$ wrap $\left\} \cup\left(P_{2} \cup\right.\right.$ wrap $\left.B_{2}\right)$
using NOTIN by auto
have 5: $\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\}) \cap\left(P_{2} \cup\right.$ wrap $B_{2} \cup$ wrap $\left.\{u\}\right)=\{ \}$
using invrules(5) NOTIN unfolding wrap-def by force
from $i n v_{1} I[O F 12345]$ show ?thesis. qed

A simplified version of the bin packing algorithm proposed in the article. It serves as an introduction into the approach taken, and, while it does not provide the desired approximation factor, it does ensure that $P$ is a correct solution of the bin packing problem.

```
lemma simple-bp-correct:
\(V A R S P P_{1} P_{2} B_{1} B_{2} V u\)
    \{True \(\}\)
    \(P_{1}:=\{ \} ; P_{2}:=\{ \} ; B_{1}:=\{ \} ; B_{2}:=\{ \} ; V:=U ;\)
    WHILE \(V \cap S \neq\{ \}\) INV \(\left\{\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2} V\right\} D O\)
        \(u:=(\) SOME \(u . u \in V) ; V:=V-\{u\} ;\)
        IF \(W\left(B_{1}\right)+w(u) \leq c\)
        THEN \(B_{1}:=B_{1} \cup\{u\}\)
        ELSE IF \(W\left(B_{2}\right)+w(u) \leq c\)
            THEN \(B_{2}:=B_{2} \cup\{u\}\)
            \(E L S E P_{2}:=P_{2} \cup\) wrap \(B_{2} ; B_{2}:=\{u\} F I ;\)
            \(P_{1}:=P_{1} \cup\) wrap \(B_{1} ; B_{1}:=\{ \} F I\)
    \(O D\);
    \(P:=P_{1} \cup\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(B_{2} \cup\{\{v\} \mid v . v \in V\}\)
    \(\{b p P\}\)
proof (vcg, goal-cases)
```

```
    case (1 P P Pr P
    show ?case
    unfolding bp-def partition-on-def pairwise-def wrap-def inv1-def
    using weight by auto
next
    case (2 P P P
    then have INV:inv1}\mp@subsup{P}{1}{}\mp@subsup{P}{2}{}\mp@subsup{B}{1}{\prime}\mp@subsup{B}{2}{}V.
    from 2 have V\not={} by blast
    then have IN:(SOME u. u\inV)\inV by (simp add: some-in-eq)
    from inv -stepA[OF INV IN] inv_-stepB[OF INV IN] inv -stepC[OF INV IN]
    show ?case by blast
next
```



```
    then show ?case unfolding inv -def by blast
qed
```


### 5.2.2 Lower Bounds for the Bin Packing Problem

lemma bp-bins-finite [simp]:
assumes bp $P$
shows $\forall B \in P$. finite $B$
using $b p E(3)[O F a s s m s] ~ U$-Finite by (meson Sup-upper finite-subset)
lemma bp-sol-finite [simp]:
assumes bp $P$
shows finite $P$
using $b p E(3)[O F a s s m s]$-Finite by (simp add: finite-UnionD)
If $P$ is a solution of the bin packing problem, then no bin in $P$ may contain more than one large object.

```
lemma only-one-L-per-bin:
    assumes \(b p P B \in P\)
    shows \(\forall x \in B . \forall y \in B . x \neq y \longrightarrow x \notin L \vee y \notin L\)
proof (rule ccontr, simp)
    assume \(\exists x \in B . \exists y \in B . x \neq y \wedge x \in L \wedge y \in L\)
    then obtain \(x y\) where \(*: x \in B y \in B x \neq y x \in L y \in L\) by blast
    then have \(c<w x+w y\) using \(L\)-def \(S\)-def by force
    have finite \(B\) using assms by simp
    have \(y \in B-\{x\}\) using \(*(2,3)\) by blast
    have \(W B=W(B-\{x\})+w x\)
        using \(*(1)\) <finite \(B\) by (simp add: sum.remove)
    also have \(\ldots=W(B-\{x\}-\{y\})+w x+w y\)
        using \(\langle y \in B-\{x\}\rangle\langle\) finite \(B\rangle\) by (simp add: sum.remove)
    finally have \(*: W B=W(B-\{x\}-\{y\})+w x+w y\).
    have \(\forall u \in B .0<w u\) using \(b p E(3)[O F \operatorname{assms}(1)]\) assms(2) weight by blast
    then have \(0 \leq W(B-\{x\}-\{y\})\) by (smt DiffD1 sum-nonneg)
    with \(*\) have \(c<W B\) using \(\langle c<w x+w y\rangle\) by simp
    then show False using bpE(4)[OF assms(1)] assms(2) by fastforce
qed
```

If $P$ is a solution of the bin packing problem, then the amount of large objects is a lower bound for the amount of bins in P .

```
lemma L-lower-bound-card:
    assumes \(b p P\)
    shows card \(L \leq\) card \(P\)
proof -
    have \(\forall x \in L . \exists B \in P . x \in B\)
        using \(b p E\) (3)[OF assms] L-def by blast
    then obtain \(f\) where \(f\)-def: \(\forall u \in L . u \in f u \wedge f u \in P\) by metis
    then have inj-on \(f L\)
        unfolding inj-on-def using only-one-L-per-bin[OF assms] by blast
    then have card-eq: card \(L=\operatorname{card}\left(f^{\prime} L\right)\) by (simp add: card-image)
    have \(f\) ' \(L \subseteq P\) using \(f\)-def by blast
    moreover have finite \(P\) using assms by simp
    ultimately have card \(\left(f^{\prime} L\right) \leq\) card \(P\) by (simp add: card-mono)
    then show ?thesis unfolding card-eq.
qed
```

If $P$ is a solution of the bin packing problem, then the amount of bins of a subset of P in which every bin contains a large object is a lower bound on the amount of large objects.
lemma subset-bp-card:
assumes $b p P M \subseteq P \forall B \in M . B \cap L \neq\{ \}$
shows card $M \leq$ card $L$
proof -
have $\forall B \in M . \exists u \in L . u \in B$ using assms(3) by fast
then have $\exists f . \forall B \in M . f B \in L \wedge f B \in B$ by metis
then obtain $f$ where $f$-def: $\forall B \in M . f B \in L \wedge f B \in B$..
have inj-on $f M$
proof (rule ccontr)
assume $\neg \operatorname{inj-on~f~} M$
then have $\exists x \in M . \exists y \in M . x \neq y \wedge f x=f y$ unfolding inj-on-def by blast
then obtain $x y$ where $*: x \in M y \in M x \neq y f x=f y$ by blast
then have $\exists u . u \in x \wedge u \in y$ using $f$-def by metis
then have $x \cap y \neq\{ \}$ by blast
moreover have pairwise disjnt $M$ using pairwise-subset[OF bpE(1)[OF assms(1)] $\operatorname{assms}(2)]$.
ultimately show False using * unfolding pairwise-def disjnt-def by simp qed
moreover have finite $L$ using $L$-def $U$-Finite by blast
moreover have $f$ ' $M \subseteq L$ using $f$-def by blast
ultimately show ?thesis using card-inj-on-le by blast qed

If $P$ is a correct solution of the bin packing problem, $i n v_{1}$ holds for the partial solution, and every bin in $P_{1} \cup$ wrap $B_{1}$ contains a large object, then the amount of bins in $P_{1} \cup$ wrap $B_{1} \cup\{\{v\} \mid v . v \in V \cap L\}$ is a lower bound for the amount of bins in $P$.
lemma L-bins-lower-bound-card:


```
    shows card ( }\mp@subsup{P}{1}{}\cup\mathrm{ wrap B}\mp@subsup{B}{1}{}\cup{{v}|v.v\inV\capL})\leq\operatorname{card}
proof -
    note invrules = inv 
    have }\forallB\in{{v}|v.v\inV\capL}. B\capL\not={} by blas
    with assms(3) have
    P
|v.v\inV}
    \forallB\in P 
    from subset-bp-card[OF invrules(1) this] show ?thesis
    using L-lower-bound-card[OF assms(1)] by linarith
qed
```

If $P$ is a correct solution of the bin packing problem, then the sum of the weights of the objects is equal to the sum of the weights of the bins in $P$.

```
lemma sum-Un-eq-sum-sum:
    assumes \(b p P\)
    shows \(\left(\sum u \in U . w u\right)=\left(\sum B \in P . W B\right)\)
proof -
    have FINITE: \(\forall B \in P\). finite \(B\) using assms by simp
    have DISJNT: \(\forall A \in P . \forall B \in P . A \neq B \longrightarrow A \cap B=\{ \}\)
        using \(b p E(1)[O F\) assms \(]\) unfolding pairwise-def disjnt-def.
    have \(\left(\sum u \in(\bigcup P) . w u\right)=\left(\sum B \in P . W B\right)\)
        using sum.Union-disjoint [OF FINITE DISJNT] by auto
    then show ?thesis unfolding \(b p E(3)[O F\) assms \(]\).
qed
```

If $P$ is a correct solution of the bin packing problem, then the sum of the weights of the items is a lower bound of amount of bins in $P$ multiplied by their maximum capacity.

```
lemma sum-lower-bound-card:
    assumes \(b p P\)
    shows \(\left(\sum u \in U . w u\right) \leq c * \operatorname{card} P\)
proof -
    have \(*: \forall B \in P .0<W B \wedge W B \leq c\)
        using bpE(2-4)[OF assms] weight by (metis UnionI assms bp-bins-finite
sum-pos)
    have \(\left(\sum u \in U . w u\right)=\left(\sum B \in P . W B\right)\)
        using sum-Un-eq-sum-sum [OF assms].
    also have \(\ldots \leq\left(\sum B \in P . c\right)\) using sum-mono \(*\) by fastforce
    also have \(\ldots=c * \operatorname{card} P\) by simp
    finally show?thesis .
qed
lemma \(b p-N E\) :
    assumes \(b p P\)
    shows \(P \neq\{ \}\)
    using \(U-N E\) bpE(3)[OF assms] by blast
```

```
lemma sum-Un-ge:
    fixes f :: - # real
    assumes finite M finite N}\forallB\inM\cupN.0<f
    shows sum fM\leq\operatorname{sum f(M\cupN)}
proof -
    have 0\leqsumfN-\operatorname{sum}f(M\capN)
        using assms by (smt DiffD1 inf.cobounded2 UnCI sum-mono2)
    then have sum fM\leq\operatorname{sum}fM+\operatorname{sumfN-\operatorname{sumf(M\capN)}}\mathbf{f}|
        by simp
    also have ... = sumf(M\cupN)
        using sum-Un[OF assms(1,2), symmetric].
    finally show ?thesis .
qed
```

If bij-exists holds, one can obtain a function which is bijective between the bins in $P$ and the objects in $V$ such that an object returned by the function would cause the bin to exceed its capacity.

```
definition bij-exists :: 'a set set }=>\mathrm{ 'a set }=>\mathrm{ bool where
    bij-exists P V = (\existsf.bij-betw f P V ^( }\forallB\inP.WB+w(fB)>c)
```

If $P$ is a functionally correct solution of the bin packing problem, $i n v_{1}$ holds for the partial solution, and such a bijective function exists between the bins in $P_{1}$ and the objects in $P_{2} \cup$ wrap $B_{2}$, the following strict lower bound can be shown:

```
lemma \(P_{1}\)-lower-bound-card:
    assumes bp \(P\) inv \(P_{1} P_{2} B_{1} B_{2} V\) bij-exists \(P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)\)
    shows card \(P_{1}+1 \leq\) card \(P\)
proof (cases \(\left\langle P_{1}=\{ \}\right\rangle\) )
    case True
    have finite \(P\) using \(\operatorname{assms}(1)\) by simp
    then have \(1 \leq \operatorname{card} P\) using bp-NE[OF assms(1)]
    by (metis Nat.add-0-right Suc-diff-1 Suc-le-mono card-gt-0-iff le0 mult-Suc-right
nat-mult-1)
    then show? ?thesis unfolding True by simp
next
    note invrules \(=\operatorname{inv}_{1} E[\) OF assms(2) \(]\)
    case False
    obtain \(f\) where \(f\)-def: bij-betw f \(P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right) \forall B \in P_{1}\). W \(B+w(f\)
B) \(>c\)
            using assms(3) unfolding bij-exists-def by blast
    have FINITE: finite \(P_{1}\) finite \(\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\) finite \(\left(P_{1} \cup P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\) finite
(wrap \(B_{1} \cup\{\{v\} \mid v . v \in V\}\) )
    using \(\operatorname{inv}_{1} E(1)[O F \operatorname{assms}(2)]\) bp-sol-finite by blast+
    have \(F: \forall B \in P_{2} \cup\) wrap \(B_{2}\). finite \(B\) using invrules(1) by simp
    have \(D: \forall A \in P_{2} \cup\) wrap \(B_{2} . \forall B \in P_{2} \cup\) wrap \(B_{2} . A \neq B \longrightarrow A \cap B=\{ \}\)
    using \(b p E(1)[O F\) invrules(1)] unfolding pairwise-def disjnt-def by auto
    have sum-eq: \(W\left(\cup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)=\left(\sum B \in P_{2} \cup\right.\) wrap \(B_{2}\). W B \()\)
```

using sum.Union-disjoint $[$ OF F D] by auto
have $\forall B \in P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V\} .0<W B$
using $b p E(2,3)[O F$ invrules(1)] weight by (metis (no-types, lifting) UnionI bp-bins-finite invrules(1) sum-pos)
then have $\left(\sum B \in P_{1} \cup P_{2} \cup\right.$ wrap $\left.B_{2} . W B\right) \leq\left(\sum B \in P_{1} \cup P_{2} \cup\right.$ wrap $B_{2}$ $\cup\left(\right.$ wrap $\left.\left.B_{1} \cup\{\{v\} \mid v . v \in V\}\right) . W B\right)$
using sum-Un-ge $[\operatorname{OF} \operatorname{FINITE}(3,4)$, of $W]$ by blast
also have $\ldots=\left(\sum B \in P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V\} . W$ $B)$ by (smt Un-assoc Un-commute)
also have $\ldots=W U$ using sum-Un-eq-sum-sum [OF invrules(1), symmetric].
finally have $*:\left(\sum B \in P_{1} \cup P_{2} \cup\right.$ wrap $\left.B_{2} . W B\right) \leq W U$.

- This follows from the fourth and final additional conjunct of $i n v_{1}$ and is necessary to combine the sums of the bins of the two partial solutions. This does not inherently follow from the union being a correct solution, as this need not be the case if $P_{1}$ and $P_{2} \cup$ wrap $B_{2}$ happened to be equal.
have DISJNT: $P_{1} \cap\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)=\{ \}$ using invrules(5) by blast
- This part of the proof is based on the proof on page 72 of the article [2].
have $c * \operatorname{card} P_{1}=\left(\sum B \in P_{1} . c\right)$ by $\operatorname{simp}$
also have $\ldots<\left(\sum B \in P_{1}\right.$. W $\left.B+w(f B)\right)$
using $f$-def(2) sum-strict-mono[OF FINITE(1) False] by fastforce
also have $\ldots=\left(\sum B \in P_{1} . W B\right)+\left(\sum B \in P_{1} . w(f B)\right)$
by (simp add: Groups-Big.comm-monoid-add-class.sum.distrib)
also have $\ldots=\left(\sum B \in P_{1} . W B\right)+W\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ unfolding sum.reindex-bij-betw $[\operatorname{OF} f-\operatorname{def}(1)$, of $w]$..
also have $\ldots=\left(\sum B \in P_{1}\right.$. W $\left.B\right)+\left(\sum B \in P_{2} \cup\right.$ wrap $B_{2}$. W $\left.B\right)$ unfolding sum-eq ..
also have $\ldots=\left(\sum B \in P_{1} \cup P_{2} \cup\right.$ wrap $\left.B_{2} . W B\right)$ using sum.union-disjoint $[O F$ FINITE (1,2) DISJNT, of W] by (simp add: Un-assoc)
also have $\ldots \leq\left(\sum u \in U . w u\right)$ using * .
also have $\ldots \leq c *$ card $P$ using sum-lower-bound-card[OF assms(1)].
finally show? ?thesis by (meson discrete nat-mult-less-cancel-disj of-nat-less-imp-less) qed

As card $($ wrap ? B $) \leq 1$ holds, it follows that the amount of bins in $P_{1}$ $\cup$ wrap $B_{1}$ are a lower bound for the amount of bins in $P$.
lemma $P_{1}$ - $B_{1}$-lower-bound-card:
assumes bp $P$ inv $P_{1} P_{2} B_{1} B_{2} V$ bij-exists $P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$
shows card $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right) \leq \operatorname{card} P$
proof -
have card $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right) \leq \operatorname{card} P_{1}+\operatorname{card}\left(\right.$ wrap $\left.B_{1}\right)$
using card-Un-le by blast
also have $\ldots \leq$ card $P_{1}+1$ using wrap-card by simp
also have $\ldots \leq$ card $P$ using $P_{1}$-lower-bound-card[OF assms].
finally show ?thesis.
qed
If $i n v_{1}$ holds, there are at most half as many bins in $P_{2}$ as there are
objects in $P_{2}$ ，and we can again obtain a bijective function between the bins in $P_{1}$ and the objects of the second partial solution，then the amount of bins in the second partial solution are a strict lower bound for half the bins of the first partial solution．

```
lemma \(P_{2}-B_{2}\)-lower-bound- \(P_{1}\) :
    assumes \(\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2} V 2 *\) card \(P_{2} \leq\) card \(\left(\bigcup P_{2}\right)\) bij-exists \(P_{1}\left(\bigcup\left(P_{2}\right.\right.\)
\(\cup\) wrap \(\left.B_{2}\right)\) )
    shows \(2 * \operatorname{card}\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right) \leq \operatorname{card} P_{1}+1\)
proof -
    note invrules \(=i n v_{1} E[O F \operatorname{assms}(1)]\) and bprules \(=b p E[O F \operatorname{invrules}(1)]\)
    have pairwise disjnt \(\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\)
        using bprules(1) pairwise-subset by blast
    moreover have \(B_{2} \notin P_{2}\) using invrules(4) by simp
    ultimately have DISJNT: \(\bigcup P_{2} \cap B_{2}=\{ \}\)
        by (auto, metis (no-types, opaque-lifting) sup-bot.right-neutral Un-insert-right
disjnt-iff \(m k\)-disjoint-insert pairwise-insert wrap-Un)
    have finite \(\left(\bigcup P_{2}\right)\) using \(U\)-Finite bprules(3) by auto
    have finite \(B_{2}\) using bp-bins-finite[OF invrules(1)] wrap-not-empty by blast
    have finite \(P_{2}\) finite (wrap \(B_{2}\) ) using bp-sol-finite[OF invrules(1)] by blast+
    have DISJNT2: \(P_{2} \cap\) wrap \(B_{2}=\{ \}\) unfolding wrap-def using \(\left\langle B_{2} \notin P_{2}\right\rangle\) by
auto
    have card \(\left(\right.\) wrap \(\left.B_{2}\right) \leq \operatorname{card} B_{2}\)
    proof (cases \(\left\langle B_{2}=\{ \}\right\rangle\) )
        case False
        then have \(1 \leq\) card \(B_{2}\) by (simp add: leI〈finite \(B_{2}\) )
        then show ?thesis using wrap-card[of \(B_{2}\) ] by linarith
    qed \(\operatorname{simp}\)
```

－This part of the proof is based on the proof on page 73 of the article［2］．
from assms（2）have 2＊card $P_{2}+2 * \operatorname{card}\left(\right.$ wrap $\left.B_{2}\right) \leq \operatorname{card}\left(\bigcup P_{2}\right)+\operatorname{card}$ $\left(\right.$ wrap $\left.B_{2}\right)+1$
using wrap－card［of $B_{2}$ ］by linarith
then have $2 *\left(\operatorname{card} P_{2}+\operatorname{card}\left(\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \leq \operatorname{card}\left(\bigcup P_{2}\right)+\operatorname{card} B_{2}+1$ using＜card（wrap $B_{2}$ ）$\leq$ card $B_{2}$ 〉 by simp
then have 2 $*\left(\operatorname{card}\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \leq \operatorname{card}\left(\bigcup P_{2} \cup B_{2}\right)+1$ using card－Un－disjoint $\left[O F<\right.$ finite $\left.\left(\bigcup P_{2}\right)\right\rangle\left\langle\right.$ finite $\left.B_{2}\right\rangle$ DISJNT］ and card－Un－disjoint $\left[O F\left\langle\right.\right.$ finite $\left.P_{2}\right\rangle\left\langle\right.$ finite（wrap $B_{2}$ ）〉DISJNT2］by argo
then have $2 *\left(\operatorname{card}\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \leq \operatorname{card}\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)+1$ by（cases $\left\langle B_{2}=\{ \}\right\rangle$ ）（auto simp：Un－commute）
then show $2 *\left(\operatorname{card}\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \leq \operatorname{card} P_{1}+1$
using assms（3）bij－betw－same－card unfolding bij－exists－def by metis qed

### 5.2.3 Proving the Approximation Factor

We define $i n v_{2}$ as it is defined in the article. These conjuncts allow us to prove the desired approximation factor.
definition inv $_{2}::$ 'a set set $\Rightarrow$ 'a set set $\Rightarrow$ 'a set $\Rightarrow$ 'a set $\Rightarrow$ ' $a$ set $\Rightarrow$ bool where $i n v_{2} \quad P_{1} \quad P_{2} \quad B_{1} \quad B_{2} V \longleftrightarrow i n v_{1} P_{1} P_{2} B_{1} B_{2} V-i n v_{1}$ holds for the partial solution

$$
\wedge\left(V \cap L \neq\{ \} \longrightarrow\left(\forall B \in P_{1} \cup \operatorname{wrap} B_{1} . B \cap L \neq\{ \}\right)\right)-
$$

If there are still large objects left, then every bin of the first partial solution must contain a large object
$\wedge$ bij-exists $P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ - There exists a bijective function between the bins of the first partial solution and the objects of the second one

$$
\wedge\left(2 * \operatorname{card} P_{2} \leq \operatorname{card}\left(\bigcup P_{2}\right)\right) \text { - There are at most twice as }
$$ many bins in $P_{2}$ as there are objects in $P_{2}$

```
lemma \(i n v_{2} E\) :
    assumes \(i n v_{2} P_{1} P_{2} B_{1} B_{2} V\)
    shows \(\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2} V\)
        and \(V \cap L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup\) wrap \(B_{1} . B \cap L \neq\{ \}\)
        and bij-exists \(P_{1}\left(\cup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)\)
        and \(2 * \operatorname{card} P_{2} \leq \operatorname{card}\left(\bigcup P_{2}\right)\)
    using assms unfolding \(i n v_{2}\)-def by blast+
lemma \(i n v_{2} I\) :
    assumes \(\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2} V\)
        and \(V \cap L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup\) wrap \(B_{1} . B \cap L \neq\{ \}\)
        and bij-exists \(P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)\)
        and \(2 *\) card \(P_{2} \leq \operatorname{card}\left(\bigcup P_{2}\right)\)
    shows \(i n v_{2} P_{1} P_{2} B_{1} B_{2} V\)
    using assms unfolding inv \(v_{2}\)-def by blast
```

If $P$ is a correct solution of the bin packing problem, $i n v_{2}$ holds for the partial solution, and there are no more small objects left to be distributed, then the amount of bins of the partial solution is no larger than $3 / 2$ of the amount of bins in $P$. This proof strongly follows the proof in Theorem 4.1 of the article [2].

```
lemma bin-packing-lower-bound-card:
    assumes \(V \cap S=\{ \} i n v_{2} P_{1} P_{2} B_{1} B_{2} V b p P\)
    shows card \(\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2} \cup\{\{v\} \mid v . v \in V\}\right) \leq 3 / 2 *\) card
\(P\)
proof (cases \(\langle V=\{ \}\rangle\) )
    note invrules \(=\) inv \(_{2} E[\) OF assms(2) \(]\)
    case True
    then have card \(\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2} \cup\{\{v\} \mid v . v \in V\}\right)\)
                \(=\operatorname{card}\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2}\right)\) by simp
    also have \(\ldots \leq \operatorname{card}\left(P_{1} \cup\right.\) wrap \(\left.B_{1}\right)+\operatorname{card}\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\)
    using card-Un-le[of \(\left\langle P_{1} \cup\right.\) wrap \(\left.\left.B_{1}\right\rangle\right]\) by (simp add: Un-assoc)
    also have \(\ldots \leq \operatorname{card} P+\operatorname{card}\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\)
```

using $P_{1}$ - $B_{1}$-lower-bound-card $[$ OF assms(3) invrules(1,3)] by simp
also have $\ldots \leq \operatorname{card} P+\operatorname{card} P / 2$
using $P_{2}$ - $B_{2}$-lower-bound- $P_{1}[$ OF invrules $(1,4,3)]$
and $P_{1}$-lower-bound-card $[$ OF $\operatorname{assms}(3) \operatorname{invrules}(1,3)]$ by linarith
finally show ?thesis by linarith

## next

note invrules $=i n v_{2} E[O F$ assms(2)]
case False
have $U=S \cup L$ using $S$-def $L$-def by blast
then have $*: V=V \cap L$
using $b p E(3)\left[O F \operatorname{inv}_{1} E(1)[O F\right.$ invrules (1) $\left.]\right]$
and assms(1) by blast
with False have $N E: V \cap L \neq\{ \}$ by simp
have card $\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $\left.B_{2} \cup\{\{v\} \mid v . v \in V\}\right)$ $=\operatorname{card}\left(P_{1} \cup\right.$ wrap $B_{1} \cup\{\{v\} \mid v . v \in V \cap L\} \cup P_{2} \cup$ wrap $\left.B_{2}\right)$
using * by (simp add: Un-commute Un-assoc)
also have $\ldots \leq \operatorname{card}\left(P_{1} \cup\right.$ wrap $\left.B_{1} \cup\{\{v\} \mid v . v \in V \cap L\}\right)+\operatorname{card}\left(P_{2} \cup\right.$ wrap $B_{2}$ )
using card-Un-le[of $\left\langle P_{1} \cup\right.$ wrap $\left.\left.B_{1} \cup\{\{v\} \mid v . v \in V \cap L\}\right\rangle\right]$ by (simp add: Un-assoc)
also have $\ldots \leq \operatorname{card} P+\operatorname{card}\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)$
using L-bins-lower-bound-card[OF assms(3) invrules(1) invrules(2)[OF NE]]
by linarith
also have $\ldots \leq \operatorname{card} P+\operatorname{card} P / 2$
using $P_{2}$ - $B_{2}$-lower-bound- $P_{1}[$ OF invrules $(1,4,3)]$
and $P_{1}$-lower-bound-card $[$ OF assms(3) invrules (1,3)] by linarith
finally show ?thesis by linarith qed

We define $i n v_{3}$ as it is defined in the article. This final conjunct allows us to prove that the invariant will be maintained by the algorithm.
definition inv $_{3}::$ ' $a$ set set $\Rightarrow$ 'a set set $\Rightarrow$ 'a set $\Rightarrow{ }^{\prime}$ 'a set $\Rightarrow{ }^{\prime}$ 'a set $\Rightarrow$ bool where $\operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} V \longleftrightarrow i n v_{2} P_{1} P_{2} B_{1} B_{2} V \wedge B_{2} \subseteq S$
lemma $i n v_{3} E$ :
assumes $i n v_{3} \quad P_{1} P_{2} B_{1} B_{2} V$
shows inv $P_{1} P_{2} B_{1} B_{2} V$ and $B_{2} \subseteq S$
using assms unfolding inv $v_{3}$-def by blast+
lemma $i n v_{3} I$ :
assumes $i n v_{2} P_{1} P_{2} B_{1} B_{2} V$ and $B_{2} \subseteq S$
shows $i n v_{3} P_{1} P_{2} B_{1} B_{2} V$
using assms unfolding $i n v_{3}$-def by blast
lemma loop-init:
$\operatorname{inv}_{3}\{ \}\{ \}\{ \}\{ \} U$
proof -
have $*: \operatorname{inv_{1}}\{ \}\{ \}\{ \}\{ \} U$
unfolding bp-def partition-on-def pairwise-def wrap-def inv $v_{1}$-def
using weight by auto
have bij-exists $\}(\cup(\} \cup$ wrap $\}))$
using bij-betwI' unfolding bij-exists-def by fastforce
from $i n v_{2} I[O F *-t h i s]$ have $i n v_{2}\{ \}\{ \}\{ \}\{ \} U$ by auto
from inv $_{3} I[$ OF this $]$ show ?thesis by blast
qed
If $B_{1}$ is empty and there are no large objects left, then $i n v_{3}$ will be maintained if $B_{1}$ is initialized with $u \in V \cap S$.

```
lemma loop-step \(A\) :
    assumes \(i n v_{3} P_{1} P_{2} B_{1} B_{2} V B_{1}=\{ \} V \cap L=\{ \} u \in V \cap S\)
    shows \(\operatorname{inv}_{3} P_{1} P_{2}\{u\} B_{2}(V-\{u\})\)
proof -
    note invrules \(=\operatorname{inv}_{2} E\left[O F\right.\) inv \(_{3} E(1)[O F\) assms(1)]]
    have WEIGHT: \(W B_{1}+w u \leq c\) using \(S\)-def \(\operatorname{assms}(2,4)\) by \(\operatorname{simp}\)
    from \(\operatorname{assms}(4)\) have \(u \in V\) by blast
    from inv \(_{1}\)-step \(A\left[O F\right.\) invrules(1) this WEIGHT] assms(2) have 1: inv \({ }_{1} P_{1} P_{2}\)
\(\{u\} B_{2}(V-\{u\})\) by \(\operatorname{simp}\)
    have 2: \((V-\{u\}) \cap L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup \operatorname{wrap}\{u\} . B \cap L \neq\{ \}\) using
assms(3) by blast
    from inv \(I\left[O F 1\right.\) 2] invrules have inv \(_{2} P_{1} P_{2}\{u\} B_{2}(V-\{u\})\) by blast
    from inv \(_{3} I\left[\right.\) OF this] show ?thesis using inv \(_{3} E(2)[\) OF \(\operatorname{assms}(1)]\).
qed
```

If $B_{1}$ is empty and there are large objects left, then $i n v_{3}$ will be maintained if $B_{1}$ is initialized with $u \in V \cap L$.
lemma loop-step $B$ :
assumes $\operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} V B_{1}=\{ \} u \in V \cap L$
shows $i n v_{3} P_{1} P_{2}\{u\} B_{2}(V-\{u\})$
proof -
note invrules $=\operatorname{inv}_{2} E\left[O F\right.$ inv $_{3} E(1)[O F$ assms(1)]]
have WEIGHT: W $B_{1}+w u \leq c$ using L-def weight assms(2,3) by simp
from $\operatorname{assms}(3)$ have $u \in V$ by blast
from inv $\operatorname{inch}_{1}$ step $A\left[O F\right.$ invrules (1) this WEIGHT] $\operatorname{assms(2)}$ have 1: inv $P_{1} P_{1} P_{2}$ $\{u\} B_{2}(V-\{u\})$ by simp
have $\forall B \in P_{1} . B \cap L \neq\{ \}$ using assms(3) invrules(2) by blast
then have 2: $(V-\{u\}) \cap L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup$ wrap $\{u\}$. $B \cap L \neq\{ \}$
using assms(3) by (metis Int-iff UnE empty-iff insertE singletonI wrap-not-empty)
from inv $_{2} I\left[\right.$ OF 1 2] invrules have $\mathrm{inv}_{2} P_{1} P_{2}\{u\} B_{2}(V-\{u\})$ by blast
from $i n v_{3} I[O F$ this $]$ show ?thesis using inv ${ }_{3} E(2)[O F \operatorname{assms}(1)]$.
qed
If $B_{1}$ is not empty and $u \in V \cap S$ does not exceed its maximum capacity, then $i n v_{3}$ will be maintained if $B_{1}$ and $\{u\}$ are replaced with $B_{1} \cup\{u\}$.

```
lemma loop-step \(C\) :
    assumes \(\operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} V B_{1} \neq\{ \} u \in V \cap S W B_{1}+w(u) \leq c\)
    shows \(\operatorname{inv}_{3} P_{1} P_{2}\left(B_{1} \cup\{u\}\right) B_{2}(V-\{u\})\)
proof -
    note invrules \(=\operatorname{inv}_{2} E\left[O F \operatorname{inv}_{3} E(1)[O F \operatorname{assms}(1)]\right]\)
```

from assms(3) have $u \in V$ by blast
from inv $\mathrm{v}_{1}$-step $A\left[O F \operatorname{invrules}(1)\right.$ this assms(4)] have 1: inv ${ }_{1} P_{1} P_{2}\left(B_{1} \cup\{u\}\right)$ $B_{2}(V-\{u\})$.
have $(V-\{u\}) \cap L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup$ wrap $B_{1} . B \cap L \neq\{ \}$ using invrules(2) by blast
then have 2: $(V-\{u\}) \cap L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup$ wrap $\left(B_{1} \cup\{u\}\right) . B \cap L \neq$ \{\}
by (metis Int-commute Un-empty-right Un-insert-right assms(2) disjoint-insert(2) insert-iff wrap-not-empty)
from inv $_{2} I[O F 12]$ invrules have inv $_{2} P_{1} P_{2}\left(B_{1} \cup\{u\}\right) B_{2}(V-\{u\})$ by blast
from $i n v_{3} I[$ OF this $]$ show ?thesis using inv $_{3} E($ 2) $[$ OF assms(1)]. qed

If $B_{1}$ is not empty and $u \in V \cap S$ does exceed its maximum capacity but not the capacity of $B_{2}$, then $i n v_{3}$ will be maintained if $B_{1}$ is added to $P_{1}$ and emptied, and $B_{2}$ and $\{u\}$ are replaced with $B_{2} \cup\{u\}$.
lemma loop-stepD:
assumes $i n v_{3} P_{1} P_{2} B_{1} B_{2} V B_{1} \neq\{ \} u \in V \cap S W B_{1}+w(u)>c W B_{2}+$ $w(u) \leq c$
shows $\operatorname{inv}_{3}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right) P_{2}\{ \}\left(B_{2} \cup\{u\}\right)(V-\{u\})$
proof -
note invrules $=$ inv $_{2} E\left[O F\right.$ inv $_{3} E(1)[$ OF assms(1) $\left.]\right]$
from $\operatorname{assms}(3)$ have $u \in V$ by blast
from inv $v_{1}$-step $B\left[O F\right.$ invrules (1) this assms(5)] have 1: inv ${ }_{1}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right) P_{2}$ $\left\}\left(B_{2} \cup\{u\}\right)(V-\{u\})\right.$.
have 2: $(V-\{u\}) \cap L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup$ wrap $B_{1} \cup$ wrap $\} . B \cap L \neq\{ \}$ using invrules(2) unfolding wrap-empty by blast
from invrules(3) obtain $f$ where $f$-def: bij-betw $f P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$
$\forall B \in P_{1} . c<W B+w(f B)$ unfolding bij-exists-def by blast
have $B_{1} \notin P_{1}$ using $\operatorname{inv}_{1} E(3)[$ OF invrules(1)] by blast
have $u \notin\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ using $\operatorname{inv}_{1} E(2)[O F \operatorname{invrules}(1)] \operatorname{assms}(3)$ by blast
then have $\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.\left(B_{2} \cup\{u\}\right)\right)\right)=\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2} \cup\{\{u\}\}\right)\right)$
by (metis Sup-empty Un-assoc Union-Un-distrib ccpo-Sup-singleton wrap-empty wrap-not-empty)
also have $\ldots=\left(\cup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \cup\{u\}$ by simp
finally have $U N:\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.\left(B_{2} \cup\{u\}\right)\right)\right)=\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \cup\{u\}$.
have wrap $B_{1}=\left\{B_{1}\right\}$ using wrap-not-empty[of $\left.B_{1}\right] \operatorname{assms}(2)$ by simp
let ?f $=f\left(B_{1}:=u\right)$
have BIJ: bij-betw ?f $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.\left(B_{2} \cup\{u\}\right)\right)\right)$
unfolding wrap-empty «wrap $\left.B_{1}=\left\{B_{1}\right\}\right\rangle U N$ using $f$-def(1) $\left\langle B_{1} \notin P_{1}\right\rangle\langle u \notin$
$\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ >
by (metis (no-types, lifting) bij-betw-cong fun-upd-other fun-upd-same notIn-Un-bij-betw3)
have $c<W B_{1}+w$ (?f $B_{1}$ ) using $\operatorname{assms}(4)$ by simp
then have $\left(\forall B \in P_{1} \cup\right.$ wrap $B_{1} . c<W B+w($ ?f $\left.B)\right)$
unfolding $\left\langle\right.$ wrap $\left.B_{1}=\left\{B_{1}\right\}\right\rangle$ using $f$ - $\operatorname{def}($ (2) by simp
with $B I J$ have bij-betw ?f $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.\left(B_{2} \cup\{u\}\right)\right)\right)$

$$
\wedge\left(\forall B \in P_{1} \cup \text { wrap } B_{1} \cdot c<W B+w(\text { ?f } B)\right) \text { by blast }
$$

then have 3: bij-exists $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.\left(B_{2} \cup\{u\}\right)\right)\right)$
unfolding bij-exists-def by blast
from $\operatorname{inv}_{2} I[O F 123]$ have $\operatorname{inv}_{2}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right) P_{2}\{ \}\left(B_{2} \cup\{u\}\right)(V-\{u\})$ using invrules(4) by blast
from $\operatorname{inv}_{3} I\left[O F\right.$ this] show ?thesis using $\operatorname{inv}_{3} E(2)[O F \operatorname{assms}(1)] \operatorname{assms}(3)$ by blast
qed
If the maximum capacity of $B_{2}$ is exceeded by $u \in V \cap S$, then $B_{2}$ must contain at least two objects.

```
lemma B2-at-least-two-objects:
    assumes inv }\mp@subsup{v}{3}{}\mp@subsup{P}{1}{}\mp@subsup{P}{2}{}\mp@subsup{B}{1}{}\mp@subsup{B}{2}{}Vu\inV\capSW\mp@subsup{B}{2}{}+w(u)>
    shows 2 \leq card B2
proof (rule ccontr, simp add: not-le)
    have FINITE: finite B B using inv E E(1)[OF inv 
        by (metis (no-types,lifting) Finite-Set.finite.simps U-Finite Union-Un-distrib
bpE(3) ccpo-Sup-singleton finite-Un wrap-not-empty)
    assume card B2<2
    then consider (0) card B}\mp@subsup{B}{2}{}=0|(1)\mathrm{ card B}\mp@subsup{B}{2}{}=1\mathrm{ by linarith
    then show False proof cases
            case 0 then have B2={} using FINITE by simp
            then show ?thesis using assms(2,3) S-def by simp
    next
            case 1 then obtain v}\mathrm{ where }\mp@subsup{B}{2}{}={v
            using card-1-singletonE by auto
            with inv 3 E(2)[OF assms(1)] have 2*wv\leqc using S-def by simp
            moreover from }\langle\mp@subsup{B}{2}{}={v}\rangle\mathrm{ have W B B = wv by simp
            ultimately show ?thesis using assms(2,3)S-def by simp
    qed
qed
```

If $B_{1}$ is not empty and $u \in V \cap S$ exceeds the maximum capacity of both $B_{1}$ and $B_{2}$, then $i n v_{3}$ will be maintained if $B_{1}$ and $B_{2}$ are added to $P_{1}$ and $P_{2}$ respectively, emptied, and $B_{2}$ initialized with $u$.

```
lemma loop-stepE:
    assumes \(i n v_{3} P_{1} P_{2} B_{1} B_{2} V B_{1} \neq\{ \} u \in V \cap S W B_{1}+w(u)>c W B_{2}+\)
\(w(u)>c\)
    shows \(\operatorname{inv}_{3}\left(P_{1} \cup\right.\) wrap \(\left.B_{1}\right)\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\{ \}\{u\}(V-\{u\})\)
proof -
    note invrules \(=\operatorname{inv}_{2} E\left[O F\right.\) inv \(\left._{3} E(1)[O F \operatorname{assms}(1)]\right]\)
    from \(\operatorname{assms}(3)\) have \(u \in V\) by blast
    from inv \(\mathbf{v}_{1}\) step \(C[O F\) invrules (1) this \(]\) have 1: inv \({ }_{1}\left(P_{1} \cup\right.\) wrap \(\left.B_{1}\right)\left(P_{2} \cup\right.\) wrap
\(\left.B_{2}\right)\}\{u\}(V-\{u\})\).
    have 2: \((V-\{u\}) \cap L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup\) wrap \(B_{1} \cup\) wrap \(\} . B \cap L \neq\{ \}\)
        using invrules(2) unfolding wrap-empty by blast
```

from invrules(3) obtain $f$ where $f$-def: bij-betw $f P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ $\forall B \in P_{1} . c<W B+w(f B)$ unfolding bij-exists-def by blast have $B_{1} \notin P_{1}$ using inv $\operatorname{in}_{1} E(3)[$ OF invrules(1)] by blast
have $u \notin\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ using $\operatorname{inv}_{1} E(2)[O F \operatorname{invrules}(1)] \operatorname{assms}(3)$ by blast
have $\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $B_{2} \cup$ wrap $\left.\left.\{u\}\right)\right)=\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2} \cup\{\{u\}\}\right)\right)$ unfolding wrap-def by simp
also have $\ldots=\left(\cup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \cup\{u\}$ by simp
finally have $U N:\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $B_{2} \cup$ wrap $\left.\left.\{u\}\right)\right)=\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \cup$ $\{u\}$. have wrap $B_{1}=\left\{B_{1}\right\}$ using wrap-not-empty[of $\left.B_{1}\right]$ assms(2) by simp let ?f $=f\left(B_{1}:=u\right)$
have $B I J$ : bij-betw ?f $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $B_{2} \cup$ wrap $\left.\left.\{u\}\right)\right)$
unfolding wrap-empty «wrap $\left.B_{1}=\left\{B_{1}\right\}\right\rangle U N$ using $f$-def(1)〈 $\left.B_{1} \notin P_{1}\right\rangle\langle u \notin$
$\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ >
by (metis (no-types, lifting) bij-betw-cong fun-upd-other fun-upd-same notIn-Un-bij-betw3)
have $c<W B_{1}+w\left(? f B_{1}\right)$ using assms(4) by simp
then have $\left(\forall B \in P_{1} \cup\right.$ wrap $B_{1} . c<W B+w($ ?f $\left.B)\right)$
unfolding «wrap $B_{1}=\left\{B_{1}\right\}$ 〉 using $f$-def(2) by simp
with $B I J$ have bij-betw ?f $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $B_{2} \cup$ wrap $\left.\left.\{u\}\right)\right)$
$\wedge\left(\forall B \in P_{1} \cup\right.$ wrap $B_{1} . c<W B+w($ ?f $\left.B)\right)$ by blast
then have 3: bij-exists $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $B_{2} \cup$ wrap $\left.\left.\{u\}\right)\right)$ unfolding bij-exists-def by blast
have 4: 2 $* \operatorname{card}\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right) \leq \operatorname{card}\left(\cup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$
proof -
note bprules $=b p E\left[O F\right.$ inv $_{1} E(1)[$ OF invrules(1)] $]$
have pairwise disjnt $\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)$
using bprules(1) pairwise-subset by blast
moreover have $B_{2} \notin P_{2}$ using $\operatorname{inv}_{1} E(4)[O F$ invrules(1)] by simp
ultimately have DISJNT: $\bigcup P_{2} \cap B_{2}=\{ \}$
by (auto, metis (no-types, opaque-lifting) sup-bot.right-neutral Un-insert-right disjnt-iff mk-disjoint-insert pairwise-insert wrap-Un)
have finite $\left(\bigcup P_{2}\right)$ using $U$-Finite bprules(3) by auto
have finite $B_{2}$ using $\operatorname{inv}_{1} E(1)[O F$ invrules(1)] bp-bins-finite wrap-not-empty by blast
have $2 * \operatorname{card}\left(P_{2} \cup \operatorname{wrap} B_{2}\right) \leq 2 *\left(\operatorname{card} P_{2}+\operatorname{card}\left(\operatorname{wrap} B_{2}\right)\right)$
using card-Un-le[of $P_{2}\left\langle\right.$ wrap $\left.\left.B_{2}\right\rangle\right]$ by simp
also have $\ldots \leq 2 *$ card $P_{2}+2$ using wrap-card by auto
also have $\ldots \leq \operatorname{card}\left(\bigcup P_{2}\right)+2$ using invrules(4) by simp
also have $\ldots \leq$ card $\left(\bigcup P_{2}\right)+$ card $B_{2}$ using $B_{2}$-at-least-two-objects[OF $\operatorname{assms}(1,3,5)]$ by $\operatorname{simp}$
also have $\ldots=\operatorname{card}\left(\cup\left(P_{2} \cup\left\{B_{2}\right\}\right)\right)$ using DISJNT card-Un-disjoint $[O F$ $\left\langle\right.$ finite $\left.\left(\bigcup P_{2}\right)\right\rangle\left\langle\right.$ finite $\left.B_{2}\right\rangle$ by (simp add: Un-commute)
also have $\ldots=\operatorname{card}\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ by $\left(\right.$ cases $\left.\left\langle B_{2}=\{ \}\right\rangle\right)$ auto
finally show ?thesis.
qed
from $\operatorname{inv}_{2} I[O F 1234]$ have $\operatorname{inv}_{2}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)\{ \}\{u\}(V$ $-\{u\})$.
from inv $_{3} I[$ OF this] show ?thesis using assms(3) by blast qed

The bin packing algorithm as it is proposed in the article [2]. $P$ will not only be a correct solution of the bin packing problem, but the amount of bins will be a lower bound for $3 / 2$ of the amount of bins of any correct solution $Q$, and thus guarantee an approximation factor of $3 / 2$ for the optimum.
lemma bp-approx:
$V A R S P P_{1} P_{2} B_{1} B_{2} V u$
\{True $\}$
$P_{1}:=\{ \} ; P_{2}:=\{ \} ; B_{1}:=\{ \} ; B_{2}:=\{ \} ; V:=U ;$
WHILE $V \cap S \neq\{ \}$ INV $\left\{\operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} V\right\} D O$
IF $B_{1} \neq\{ \}$
THEN $u:=(S O M E \quad u . u \in V \cap S)$
ELSE IF $V \cap L \neq\{ \}$
THEN $u:=($ SOME $u . u \in V \cap L)$
ELSE $u:=(S O M E$ u. $u \in V \cap S) F I F I ;$
$V:=V-\{u\} ;$
IF $W\left(B_{1}\right)+w(u) \leq c$
THEN $B_{1}:=B_{1} \cup\{u\}$
ELSE IF $W\left(B_{2}\right)+w(u) \leq c$
THEN $B_{2}:=B_{2} \cup\{u\}$
$E L S E P_{2}:=P_{2} \cup$ wrap $B_{2} ; B_{2}:=\{u\} F I ;$
$P_{1}:=P_{1} \cup$ wrap $B_{1} ; B_{1}:=\{ \} F I$
$O D$;
$P:=P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V\}$
$\{b p P \wedge(\forall Q . b p Q \longrightarrow$ card $P \leq 3 / 2$ * card $Q)\}$
proof (vcg, goal-cases)
case ( $1 \begin{array}{lllll} & P & P_{1} & P_{2} & B_{1}\end{array} B_{2} V u$ )
then show? case by (simp add: loop-init)
next
case (2 $\left.P P_{1} P_{2} B_{1} B_{2} V u\right)$
then have $I N V: \operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} V .$.
let $? s=S O M E u . u \in V \cap S$
let ?l $=$ SOME $u . u \in V \cap L$
have LIN: $V \cap L \neq\{ \} \Longrightarrow ? l \in V \cap L$ using some-in-eq by metis
then have LWEIGHT: $V \cap L \neq\{ \} \Longrightarrow w ? l \leq c$ using L-def weight by blast
from 2 have $V \cap S \neq\{ \}$..
then have $I N$ : ?s $\in V \cap S$ using some-in-eq by metis
then have $w$ ?s $\leq c$ using $S$-def by simp
then show ?case
using LWEIGHT loop-stepA[OF INV - - IN] loop-stepB[OF INV - LIN]
loop-step $C[O F I N V-I N]$
and loop-step $D[O F I N V-I N]$ loop-step $E[O F I N V-I N]$ by $\left(\right.$ cases $\left\langle B_{1}=\{ \}\right\rangle$,

```
cases < V\cap }L={}>) aut
next
    case (3 P P P P P P B Bllll
    then have INV: inv 都 P}\mp@subsup{P}{2}{}\mp@subsup{B}{1}{}\mp@subsup{B}{2}{}V\mathrm{ and EMPTY:V }\capS={}\mathrm{ by blast+
    from inv }E(1)[OF inv 2E(1)[OF inv 3E(1)[OF INV]]] and bin-packing-lower-bound-card[OF
EMPTY inv E(1)[OF INV]]
    show ?case by blast
qed
end
```


### 5.3 The Full Linear Time Version of the Proposed Algorithm

Finally, we prove the Algorithm proposed on page 78 of the article [2]. This version generates the $S$ and $L$ sets beforehand and uses them directly to calculate the solution, thus removing the need for intersection operations, and ensuring linear time if we can perform insertion, removal, and selection of an element, the union of two sets, and the emptiness test in constant time [2].
locale BinPacking-Complete $=$
fixes $U$ :: ' $a$ set - A finite, non-empty set of objects
and $w::{ }^{\prime} a \Rightarrow$ real - A mapping from objects to their respective weights (positive real numbers)
and $c::$ nat - The maximum capacity of a bin (as a natural number)
assumes weight: $\forall u \in U .0<w(u) \wedge w(u) \leq c$
and $U$-Finite: finite $U$
and $U-N E: U \neq\{ \}$
begin
The correctness proofs will be identical to the ones of the simplified algorithm.
abbreviation $W::$ ' a set $\Rightarrow$ real where $W B \equiv\left(\sum u \in B . w(u)\right)$
definition $b p::$ ' $a$ set set $\Rightarrow$ bool where $b p P \longleftrightarrow$ partition-on $U P \wedge(\forall B \in P . W(B) \leq c)$
lemma $b p E$ :
assumes $b p P$
shows pairwise disjnt $P\} \notin P \bigcup P=U \forall B \in P . W(B) \leq c$
using assms unfolding bp-def partition-on-def by blast+
lemma $b p I$ :
assumes pairwise disjnt $P\} \notin P \bigcup P=U \forall B \in P . W(B) \leq c$
shows bp $P$
using assms unfolding bp-def partition-on-def by blast
definition $i n v_{1}::{ }^{\prime} a$ set set $\Rightarrow{ }^{\prime} a$ set set $\Rightarrow{ }^{\prime} a$ set $\Rightarrow{ }^{\prime} a$ set $\Rightarrow{ }^{\prime} a$ set $\Rightarrow$ bool where
$i n v_{1} P_{1} P_{2} B_{1} B_{2} V \longleftrightarrow b p\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in$ $V\})$ - A correct solution to the bin packing problem
$\wedge \bigcup\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $\left.B_{2}\right)=U-V-$ The partial solution does not contain objects that have not yet been assigned $\wedge B_{1} \notin\left(P_{1} \cup P_{2} \cup\right.$ wrap $\left.B_{2}\right)-B_{1}$ is distinct from all the other bins $\wedge B_{2} \notin\left(P_{1} \cup\right.$ wrap $\left.B_{1} \cup P_{2}\right)-B_{2}$ is distinct from all the other bins

$$
\wedge\left(P_{1} \cup \operatorname{wrap} B_{1}\right) \cap\left(P_{2} \cup \operatorname{wrap} B_{2}\right)=\{ \} \text { - The first and }
$$ second partial solutions are disjoint from each other.

```
lemma \(i n v_{1} E\) :
    assumes \(i n v_{1} P_{1} P_{2} B_{1} B_{2} V\)
    shows \(b p\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2} \cup\{\{v\} \mid v . v \in V\}\right)\)
        and \(\cup\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2}\right)=U-V\)
        and \(B_{1} \notin\left(P_{1} \cup P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\)
        and \(B_{2} \notin\left(P_{1} \cup\right.\) wrap \(\left.B_{1} \cup P_{2}\right)\)
        and \(\left(P_{1} \cup\right.\) wrap \(\left.B_{1}\right) \cap\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)=\{ \}\)
    using assms unfolding inv \(v_{1}\)-def by auto
lemma \(i n v_{1} I\) :
    assumes \(b p\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2} \cup\{\{v\} \mid v . v \in V\}\right)\)
        and \(\cup\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2}\right)=U-V\)
        and \(B_{1} \notin\left(P_{1} \cup P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\)
        and \(B_{2} \notin\left(P_{1} \cup\right.\) wrap \(\left.B_{1} \cup P_{2}\right)\)
        and \(\left(P_{1} \cup\right.\) wrap \(\left.B_{1}\right) \cap\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)=\{ \}\)
    shows \(i n v_{1} P_{1} P_{2} B_{1} B_{2} V\)
    using assms unfolding inv \(v_{1}\)-def by blast
```

lemma wrap-Un [simp]: wrap $(M \cup\{x\})=\{M \cup\{x\}\}$ unfolding wrap-def by
simp
lemma wrap-empty [simp]: wrap $\}=\{ \}$ unfolding wrap-def by simp
lemma wrap-not-empty $[$ simp $]: M \neq\{ \} \longleftrightarrow$ wrap $M=\{M\}$ unfolding wrap-def
by $\operatorname{simp}$
lemma inv $v_{1}$-step $A$ :
assumes $\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2} V u \in V W\left(B_{1}\right)+w(u) \leq c$
shows $\operatorname{inv}_{1} P_{1} P_{2}\left(B_{1} \cup\{u\}\right) B_{2}(V-\{u\})$
proof -
note invrules $=\operatorname{inv}_{1} E[$ OF assms(1) $]$ and bprules $=b p E[$ OF invrules(1) $]$

- Rule 1: Pairwise Disjoint
have NOTIN: $\forall M \in P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup \operatorname{wrap} B_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$. $u \notin M$
using invrules(2) assms(2) by blast
have $\{\{v\} \mid v . v \in V\}=\{\{u\}\} \cup\{\{v\} \mid v . v \in V-\{u\}\}$
using assms(2) by blast
then have pairwise disjnt $\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup(\{\{u\}\} \cup\{\{v\} \mid v$. $v \in V-\{u\}\}))$
using $\operatorname{bprules}(1) \operatorname{assms}(2)$ by $\operatorname{simp}$
then have pairwise disjnt (wrap $B_{1} \cup\{\{u\}\} \cup P_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v$. $v \in V-\{u\}\}$ ) by (simp add: Un-commute)
then have assm: pairwise disjnt (wrap $B_{1} \cup\{\{u\}\} \cup\left(P_{1} \cup P_{2} \cup\right.$ wrap $B_{2} \cup$ $\{\{v\} \mid v . v \in V-\{u\}\})$ ) by (simp add: Un-assoc)
have pairwise disjnt $\left(\left\{B_{1} \cup\{u\}\right\} \cup\left(P_{1} \cup P_{2} \cup\right.\right.$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-$ $\{u\}\}$ ))
proof (cases $\left.\left\langle B_{1}=\{ \}\right\rangle\right)$
case True with assm show ?thesis by simp
next
case False
with assm have assm: pairwise disjnt $\left(\left\{B_{1}\right\} \cup\{\{u\}\} \cup\left(P_{1} \cup P_{2} \cup\right.\right.$ wrap $B_{2}$ $\cup\{\{v\} \mid v . v \in V-\{u\}\}))$ by simp
from NOTIN have $\{u\} \notin P_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$ by blast
from pairwise-disjnt-Un[OF assm - this] invrules $(2,3)$ show ?thesis
using False by auto
qed
then have 1: pairwise disjnt $\left(P_{1} \cup\right.$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\}$ $\mid v . v \in V-\{u\}\})$
unfolding wrap-Un by simp
- Rule 2: No empty sets
from bprules(2) have 2: $\left\} \notin P_{1} \cup \operatorname{wrap}\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup\right.$ wrap $B_{2} \cup\{\{v\}$ $\mid v . v \in V-\{u\}\}$
unfolding wrap-def by simp
- Rule 3: Union preserved
from $\operatorname{bprules}(3)$ have $\cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{u\}\} \cup\{\{v\} \mid v . v$ $\in V-\{u\}\})=U$
using assms(2) by blast
then have 3: $\cup\left(P_{1} \cup\right.$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-$ $\{u\}\})=U$
unfolding wrap-def by force
- Rule 4: Weights below capacity
have $0<w u$ using weight assms(2) bprules(3) by blast
have finite $B_{1}$ using bprules(3) U-Finite by (cases $\left\langle B_{1}=\{ \}\right\rangle$ ) auto
then have $W\left(B_{1} \cup\{u\}\right) \leq W B_{1}+w u$ using $\langle 0<w u\rangle$ by (cases $\left.\left\langle u \in B_{1}\right\rangle\right)$
(auto simp: insert-absorb)
also have $\ldots \leq c$ using $\operatorname{assms}(3)$.
finally have $W\left(B_{1} \cup\{u\}\right) \leq c$.
then have $\forall B \in \operatorname{wrap}\left(B_{1} \cup\{u\}\right)$. W $B \leq c$ unfolding wrap-Un by blast
moreover have $\forall B \in P_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$. W $B \leq c$
using bprules(4) by blast
ultimately have $4: \forall B \in P_{1} \cup$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v$ $\in V-\{u\}\} . W B \leq c$ by blast
from $b p I[O F 1234]$ have 1: $b p\left(P_{1} \cup \operatorname{wrap}\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup\right.$ wrap $B_{2} \cup$ $\{\{v\} \mid v . v \in V-\{u\}\})$.
- Auxiliary information is preserved
have $u \in U$ using assms(2) bprules(3) by blast
then have $R$ : $U-(V-\{u\})=U-V \cup\{u\}$ by blast
have $L: \cup\left(P_{1} \cup\right.$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup$ wrap $\left.B_{2}\right)=\bigcup\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2}$
$\cup$ wrap $\left.B_{2}\right) \cup\{u\}$
unfolding wrap-def using NOTIN by auto
have 2: $\cup\left(P_{1} \cup \operatorname{wrap}\left(B_{1} \cup\{u\}\right) \cup P_{2} \cup\right.$ wrap $\left.B_{2}\right)=U-(V-\{u\})$
unfolding $L R$ invrules(2) ..
have 3: $B_{1} \cup\{u\} \notin P_{1} \cup P_{2} \cup$ wrap $B_{2}$
using NOTIN by auto
have 4: $B_{2} \notin P_{1} \cup$ wrap $\left(B_{1} \cup\{u\}\right) \cup P_{2}$
using invrules(4) NOTIN unfolding wrap-def by fastforce
have 5: $\left(P_{1} \cup\right.$ wrap $\left.\left(B_{1} \cup\{u\}\right)\right) \cap\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)=\{ \}$
using invrules(5) NOTIN unfolding wrap-Un by auto
from $i n v_{1} I[O F 12345]$ show ?thesis.
qed
lemma inv $_{1}$-step $B$ :
assumes inv $P_{1} P_{1} B_{1} B_{2} V u \in V W B_{2}+w u \leq c$
shows $\operatorname{inv}_{1}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right) P_{2}\{ \}\left(B_{2} \cup\{u\}\right)(V-\{u\})$
proof -
note invrules $=\operatorname{inv}_{1} E[$ OF assms(1) $]$ and bprules $=b p E[O F \operatorname{invrules}(1)]$
have NOTIN: $\forall M \in P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$. $u \notin M$
using invrules(2) assms(2) by blast
have $\{\{v\} \mid v . v \in V\}=\{\{u\}\} \cup\{\{v\} \mid v . v \in V-\{u\}\}$
using assms(2) by blast
then have pairwise disjnt $\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{u\}\} \cup\{\{v\} \mid v$. $v \in V-\{u\}\})$
using bprules(1) assms(2) by simp
then have assm: pairwise disjnt (wrap $B_{2} \cup\{\{u\}\} \cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ $\{\{v\} \mid v . v \in V-\{u\}\}))$
by (simp add: Un-assoc Un-commute)
have pairwise disjnt $\left(\left\{B_{2} \cup\{u\}\right\} \cup\left(P_{1} \cup\right.\right.$ wrap $B_{1} \cup P_{2} \cup\{\{v\} \mid v . v \in V-$ $\{u\}\})$ )
proof (cases $\left\langle B_{2}=\{ \}\right\rangle$ )
case True with assm show? ?thesis by simp
next
case False
with assm have assm: pairwise disjnt $\left(\left\{B_{2}\right\} \cup\{\{u\}\} \cup\left(P_{1} \cup\right.\right.$ wrap $B_{1} \cup P_{2}$ $\cup\{\{v\} \mid v . v \in V-\{u\}\}))$ by $\operatorname{simp}$
from NOTIN have $\{u\} \notin P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$ by blast
from pairwise-disjnt-Un[OF assm - this] invrules $(2,4)$ show ?thesis using False by auto
qed
then have 1: pairwise disjnt $\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2} \cup\right.$ wrap $\left(B_{2} \cup\{u\}\right)$ $\cup\{\{v\} \mid v . v \in V-\{u\}\})$
unfolding wrap-Un by simp
- Rule 2: No empty sets
from bprules(2) have 2: $\left\} \notin P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2} \cup\right.$ wrap $\left(B_{2} \cup\right.$ $\{u\}) \cup\{\{v\} \mid v . v \in V-\{u\}\}$
unfolding wrap-def by simp
- Rule 3: Union preserved
from bprules $(3)$ have $\bigcup\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{u\}\} \cup\{\{v\} \mid v . v$ $\in V-\{u\}\})=U$
using assms(2) by blast
then have 3: $\bigcup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2} \cup\right.$ wrap $\left(B_{2} \cup\{u\}\right) \cup\{\{v\} \mid v$. $v \in V-\{u\}\})=U$
unfolding wrap-def by force
- Rule 4: Weights below capacity
have $0<w u$ using weight assms(2) bprules(3) by blast
have finite $B_{2}$ using bprules(3) U-Finite by (cases $\left\langle B_{2}=\{ \}\right\rangle$ ) auto
then have $W\left(B_{2} \cup\{u\}\right) \leq W B_{2}+w u$ using $\langle 0<w u\rangle$ by (cases $\left\langle u \in B_{2}\right\rangle$ )
(auto simp: insert-absorb)
also have $\ldots \leq c$ using $\operatorname{assms}$ (3).
finally have $W\left(B_{2} \cup\{u\}\right) \leq c$.
then have $\forall B \in$ wrap $\left(B_{2} \cup\{u\}\right)$. W $B \leq c$ unfolding wrap- Un by blast
moreover have $\forall B \in P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$. W $B \leq c$ using bprules(4) by blast
ultimately have $4: \forall B \in P_{1} \cup$ wrap $B_{1} \cup \operatorname{wrap}\{ \} \cup P_{2} \cup \operatorname{wrap}\left(B_{2} \cup\{u\}\right) \cup$ $\{\{v\} \mid v . v \in V-\{u\}\} . W B \leq c$
by auto
from $b p I[O F 1234]$ have $1: b p\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\{ \} \cup P_{2} \cup$ wrap $\left(B_{2}\right.$ $\cup\{u\}) \cup\{\{v\} \mid v . v \in V-\{u\}\})$.
- Auxiliary information is preserved
have $u \in U$ using $\operatorname{assms}(2)$ bprules(3) by blast
then have $R$ : $U-(V-\{u\})=U-V \cup\{u\}$ by blast
have $L: \cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\{ \} \cup P_{2} \cup$ wrap $\left.\left(B_{2} \cup\{u\}\right)\right)=\bigcup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2} \cup\right.$ wrap $\left.B_{2}\right) \cup\{u\}$
unfolding wrap-def using NOTIN by auto
have 2: $\cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2} \cup \operatorname{wrap}\left(B_{2} \cup\{u\}\right)\right)=U-(V-$ $\{u\}$ )
unfolding $L R$ using invrules(2) by simp
have 3: $\left\} \notin P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $\left(B_{2} \cup\{u\}\right)$ using $b p E(2)[O F 1]$ by simp
have $4: B_{2} \cup\{u\} \notin P_{1} \cup$ wrap $B_{1} \cup$ wrap $\left\} \cup P_{2}\right.$
using NOTIN by auto
have 5: $\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\}) \cap\left(P_{2} \cup\right.$ wrap $\left.\left(B_{2} \cup\{u\}\right)\right)=\{ \}$
using invrules(5) NOTIN unfolding wrap-empty wrap-Un by auto
from $i n v_{1} I[$ OF 122345$]$ show ?thesis.
qed
lemma inv $_{1}-$ step $C$ :
assumes $\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2} V u \in V$
shows $\operatorname{inv}_{1}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)\{ \}\{u\}(V-\{u\})$
proof -
note invrules $=i n v_{1} E[O F \operatorname{assms}(1)]$
- Rule 1-4: Correct Bin Packing
have $P_{1} \cup$ wrap $B_{1} \cup$ wrap $\left\} \cup\left(P_{2} \cup\right.\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\{u\} \cup\{\{v\} \mid v . v \in V$ - $\{u\}\}$
$=P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{u\}\} \cup\{\{v\} \mid v . v \in V-\{u\}\}$
by (metis (no-types, lifting) Un-assoc Un-empty-right insert-not-empty wrap-empty wrap-not-empty)
also have $\ldots=P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V\}$
using assms(2) by auto
finally have $E Q: P_{1} \cup$ wrap $B_{1} \cup$ wrap $\left\} \cup\left(P_{2} \cup\right.\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\{u\} \cup$ $\{\{v\} \mid v . v \in V-\{u\}\}$

$$
=P_{1} \cup \text { wrap } B_{1} \cup P_{2} \cup \text { wrap } B_{2} \cup\{\{v\} \mid v . v \in V\} .
$$

from invrules $(1)$ have $1: b p\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\{ \} \cup\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\{u\} \cup\{\{v\} \mid v . v \in V-\{u\}\})$
unfolding $E Q$.

- Auxiliary information is preserved
have NOTIN: $\forall M \in P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in V-\{u\}\}$. $u \notin M$
using invrules(2) assms(2) by blast
have $u \in U$ using $\operatorname{assms(2)} b p E(3)[$ OF invrules(1)] by blast
then have $R$ : $U-(V-\{u\})=U-V \cup\{u\}$ by blast
have $L: \cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\{ \} \cup\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\left.\{u\}\right)=\bigcup\left(P_{1}\right.$ $\cup$ wrap $B_{1} \cup$ wrap $\left\} \cup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \cup\{u\}$
unfolding wrap-def using NOTIN by auto
have 2: $\cup\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\left\} \cup\left(P_{2} \cup\right.\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\left.\{u\}\right)=U-(V$ $-\{u\})$
unfolding $L R$ using invrules(2) by auto
have 3: $\left\} \notin P_{1} \cup\right.$ wrap $B_{1} \cup\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right) \cup$ wrap $\{u\}$ using $b p E(2)[O F 1]$ by $\operatorname{simp}$
have $4:\{u\} \notin P_{1} \cup$ wrap $B_{1} \cup$ wrap $\left\} \cup\left(P_{2} \cup\right.\right.$ wrap $\left.B_{2}\right)$
using NOTIN by auto
have 5: $\left(P_{1} \cup\right.$ wrap $B_{1} \cup$ wrap $\}) \cap\left(P_{2} \cup\right.$ wrap $B_{2} \cup$ wrap $\left.\{u\}\right)=\{ \}$
using invrules(5) NOTIN unfolding wrap-def by force
from $\operatorname{inv}_{1} I[$ OF 123 4 5] show ?thesis.
qed
From this point onward, we will require a different approach for proving lower bounds. Instead of fixing and assuming the definitions of the $S$ and $L$ sets, we will introduce the abbreviations $S_{U}$ and $L_{U}$ for any occurrences of the original $S$ and $L$ sets. The union of $S$ and $L$ can be interpreted as
$V$. As a result, occurrences of $V \cap S$ become $(S \cup L) \cap S=S$, and $V \cap L$ become $(S \cup L) \cap L=L$. Occurrences of these sets will have to be replaced appropriately.
abbreviation $S_{U}$ where

$$
S_{U} \equiv\{u \in U \cdot w u \leq c / 2\}
$$

## abbreviation $L_{U}$ where

$$
L_{U} \equiv\{u \in U \cdot c / 2<w u\}
$$

As we will remove elements from $S$ and $L$, we will only be able to show that they remain subsets of $S_{U}$ and $L_{U}$ respectively.

```
abbreviation SL where
    SL SL\equivS\subseteq S S ^L\subseteq L
lemma bp-bins-finite [simp]:
    assumes bp P
    shows }\forallB\inP\mathrm{ . finite }
    using bpE(3)[OF assms] U-Finite by (meson Sup-upper finite-subset)
lemma bp-sol-finite [simp]:
    assumes bp P
    shows finite P
    using bpE(3)[OF assms] U-Finite by (simp add: finite-UnionD)
lemma only-one-L-per-bin:
    assumes bp P B \inP
    shows }\forallx\inB.\forally\inB. x\not=y\longrightarrowx\not\in\mp@subsup{L}{U}{}\veey\not\in\mp@subsup{L}{U}{
proof (rule ccontr, simp)
```



```
wy*2
    then obtain x y where *: x\inB y\inB x\not=yx\in L LU y\in L LU by auto
    then have c<wx+wy by force
    have finite B using assms by simp
    have }y\inB-{x} using *(2,3) by blas
    have WB=W(B-{x})+wx
        using *(1) <finite B> by (simp add: sum.remove)
    also have ... =W W ( - {x}-{y})+wx+wy
        using «y \inB - {x}><finite B> by (simp add: sum.remove)
    finally have *:W B=W(B-{x}-{y})+wx+wy.
    have }\forallu\inB.0<wu\mathrm{ using bpE(3)[OF assms(1)] assms(2) weight by blast
    then have 0\leqW (B-{x} - {y}) by (smt DiffD1 sum-nonneg)
    with * have c< W B using <c< <wx+wy> by simp
    then show False using bpE(4)[OF assms(1)] assms(2) by fastforce
qed
lemma L-lower-bound-card:
    assumes bp P
    shows card LU L card P
proof -
```

```
    have }\forallx\in\mp@subsup{L}{U}{}.\existsB\inP.x\in
    using bpE(3)[OF assms] by blast
    then obtain f}\mathrm{ where f-def: }\forallu\in\mp@subsup{L}{U}{}.u\infu\wedgefu\inP by meti
    then have inj-on f L LU
        unfolding inj-on-def using only-one-L-per-bin[OF assms] by blast
    then have card-eq: card L}\mp@subsup{L}{U}{}=\operatorname{card}(\mp@subsup{f}{}{\prime}\mp@subsup{L}{U}{})\mathrm{ by (simp add: card-image)
    have f' L}\mp@subsup{L}{U}{}\subseteqP\mathrm{ using f-def by blast
    moreover have finite P using assms by simp
    ultimately have card (f' 'LU})\leq\operatorname{card P by (simp add: card-mono)
    then show ?thesis unfolding card-eq.
qed
lemma subset-bp-card:
    assumes bp PM\subseteqP\forallB\inM.B\cap邱{}
    shows card M}\leq\mathrm{ card L LU
proof -
    have }\forallB\inM.\existsu\in\mp@subsup{L}{U}{}.u\inB\mathrm{ using assms(3) by fast
    then have }\existsf.\forallB\inM.fB\in\mp@subsup{L}{U}{}\wedgefB\inB\mathrm{ by metis
    then obtain f where f-def:}\forallB\inM.fB\in\mp@subsup{L}{U}{}\wedgefB\inB.
    have inj-on f M
    proof (rule ccontr)
        assume }\neg\mathrm{ inj-on f M
        then have \existsx\inM.\existsy\inM.x\not=y^fx=fy unfolding inj-on-def by blast
        then obtain }xy\mathrm{ where *: x GMy M Mx; yfx=fy by blast
        then have }\existsu.u\inx\wedgeu\iny\mathrm{ using f-def by metis
        then have }x\capy\not={}\mathrm{ by blast
    moreover have pairwise disjnt M using pairwise-subset[OF bpE(1)[OF assms(1)]
assms(2)].
    ultimately show False using * unfolding pairwise-def disjnt-def by simp
    qed
    moreover have finite }\mp@subsup{L}{U}{}\mathrm{ using U-Finite by auto
    moreover have f'M\subseteq L
    ultimately show ?thesis using card-inj-on-le by blast
qed
lemma L-bins-lower-bound-card:
    assumes bp Pinv 1 P P P P B B B (S\cupL)\forallB\in P
        and SL-def:SLSL
    shows card ( }\mp@subsup{P}{1}{}\cup\mathrm{ wrap B}\mp@subsup{B}{1}{}\cup{{v}|v.v\inL})\leq\operatorname{card}
proof -
    note invrules = inv }E[OF\mathrm{ assms(2)]
    have }\forallB\in{{v}|v.v\inL}.B\cap\mp@subsup{L}{U}{}\not={}\mathrm{ using SL-def by blast
    with assms(3) have
        P
v\inS\cupL}
    \forallB\in\mp@subsup{P}{1}{}\cupwrap B}\mp@subsup{B}{1}{}\cup{{v}|v.v\inL}.B\cap\mp@subsup{L}{U}{}\not={}\mathrm{ by blast+
    from subset-bp-card[OF invrules(1) this] show ?thesis
        using L-lower-bound-card[OF assms(1)] by linarith
qed
```

```
lemma sum-Un-eq-sum-sum:
    assumes bp P
    shows (\sumu\inU.wu)=(\sumB\inP.WB)
proof -
    have FINITE: }\forallB\inP\mathrm{ . finite B using assms by simp
    have DISJNT: }\forallA\inP.\forallB\inP.A\not=B\longrightarrowA\capB={
        using bpE(1)[OF assms] unfolding pairwise-def disjnt-def .
    have (\sumu\in(\bigcupP).wu)=(\sumB\inP.WB)
        using sum.Union-disjoint[OF FINITE DISJNT] by auto
    then show ?thesis unfolding bpE(3)[OF assms].
qed
lemma sum-lower-bound-card:
    assumes bp P
    shows (\sumu\inU.wu)\leqc* card P
proof -
    have *:}\forallB\inP.0<WB\wedgeWB\leq
        using bpE(2-4)[OF assms] weight by (metis UnionI assms bp-bins-finite
sum-pos)
    have (\sumu\inU.wu)=(\sumB\inP.WB)
        using sum-Un-eq-sum-sum[OF assms].
    also have ... \leq(\sumB\inP.c) using sum-mono * by fastforce
    also have ... =c* card P by simp
    finally show?thesis.
qed
lemma bp-NE:
    assumes bp P
    shows P}\not={
    using U-NE bpE(3)[OF assms] by blast
lemma sum-Un-ge:
    fixes f:: - # real
    assumes finite M finite N}\forallB\inM\cupN.0<f
    shows sumf M\leqsumf(M\cupN)
proof -
    have 0\leqsumfN-\operatorname{sumf(M\capN)}
        using assms by (smt DiffD1 inf.cobounded2 UnCI sum-mono2)
    then have sumfM\leq\operatorname{sumfM}+\operatorname{sumfN-sumf(M\capN)}
        by simp
    also have ... = sumf ( }M\cupN
        using sum-Un[OF assms(1,2), symmetric].
    finally show ?thesis.
qed
definition bij-exists :: 'a set set }=>\mathrm{ 'a set }=>\mathrm{ bool where
    bij-exists P V = (\existsf.bij-betw f P V ^( }\forallB\inP.WB+w(fB)>c)
```


## lemma $P_{1}$-lower-bound-card:

assumes bp $P \operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2}(S \cup L)$ bij-exists $P_{1}\left(\cup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$
shows card $P_{1}+1 \leq$ card $P$
proof (cases $\left\langle P_{1}=\{ \}\right\rangle$ )
case True
have finite $P$ using assms(1) by simp
then have $1 \leq \operatorname{card} P$ using bp-NE[OF assms(1)]
by (metis Nat.add-0-right Suc-diff-1 Suc-le-mono card-gt-0-iff le0 mult-Suc-right nat-mult-1)
then show? ?thesis unfolding True by simp
next
note invrules $=$ inv $_{1} E[$ OF assms(2)]
case False
obtain $f$ where $f$-def: bij-betw $f P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \forall B \in P_{1}$. W $B+w(f$ B) $>c$
using assms(3) unfolding bij-exists-def by blast
have FINITE: finite $P_{1}$ finite $\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)$ finite $\left(P_{1} \cup P_{2} \cup\right.$ wrap $\left.B_{2}\right)$ finite $\left(\right.$ wrap $\left.B_{1} \cup\{\{v\} \mid v . v \in S \cup L\}\right)$
using $\operatorname{inv}_{1} E(1)[O F \operatorname{assms}(2)]$ bp-sol-finite by blast+
have $F: \forall B \in P_{2} \cup$ wrap $B_{2}$. finite $B$ using invrules(1) by simp
have $D: \forall A \in P_{2} \cup$ wrap $B_{2} . \forall B \in P_{2} \cup$ wrap $B_{2} . A \neq B \longrightarrow A \cap B=\{ \}$
using $b p E(1)[O F$ invrules(1)] unfolding pairwise-def disjnt-def by auto
have sum-eq: $W\left(\cup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)=\left(\sum B \in P_{2} \cup\right.$ wrap $B_{2}$. W B $)$
using sum.Union-disjoint $[$ OF F D] by auto
have $\forall B \in P_{1} \cup$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in S \cup L\} .0<W B$
using $b p E(2,3)[O F$ invrules(1)] weight by (metis (no-types, lifting) UnionI bp-bins-finite invrules(1) sum-pos)
then have $\left(\sum B \in P_{1} \cup P_{2} \cup\right.$ wrap $\left.B_{2} . W B\right) \leq\left(\sum B \in P_{1} \cup P_{2} \cup\right.$ wrap $B_{2}$ $\cup\left(\right.$ wrap $\left.\left.B_{1} \cup\{\{v\} \mid v . v \in S \cup L\}\right) . W B\right)$ using sum-Un-ge $[\operatorname{OF} \operatorname{FINITE}(3,4)$, of $W]$ by blast
also have $\ldots=\left(\sum B \in P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $B_{2} \cup\{\{v\} \mid v . v \in S \cup L\}$.
$W B$ ) by (smt Un-assoc Un-commute)
also have $\ldots=W U$ using sum-Un-eq-sum-sum $[$ OF invrules(1), symmetric] .
finally have $*:\left(\sum B \in P_{1} \cup P_{2} \cup\right.$ wrap $\left.B_{2} . W B\right) \leq W U$.
have DISJNT: $P_{1} \cap\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)=\{ \}$ using invrules(5) by blast

- This part of the proof is based on the proof on page 72 of the article [2].
have $c * \operatorname{card} P_{1}=\left(\sum B \in P_{1} . c\right)$ by simp
also have $\ldots<\left(\sum B \in P_{1} . W B+w(f B)\right)$
using $f$-def(2) sum-strict-mono[OF FINITE(1) False] by fastforce
also have $\ldots=\left(\sum B \in P_{1} . W B\right)+\left(\sum B \in P_{1} . w(f B)\right)$
by (simp add: Groups-Big.comm-monoid-add-class.sum.distrib)
also have $\ldots=\left(\sum B \in P_{1} . W B\right)+W\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ unfolding sum.reindex-bij-betw $[O F f$ - $\operatorname{def}(1)$, of $w]$..
also have $\ldots=\left(\sum B \in P_{1}\right.$. W $\left.B\right)+\left(\sum B \in P_{2} \cup\right.$ wrap $B_{2}$. W $\left.B\right)$ unfolding sum-eq ..
also have $\ldots=\left(\sum B \in P_{1} \cup P_{2} \cup\right.$ wrap $B_{2}$. W B) using sum.union-disjoint $[O F$

```
FINITE (1,2) DISJNT, of W] by (simp add: Un-assoc)
    also have \(\ldots \leq\left(\sum u \in U . w u\right)\) using \(*\).
    also have \(\ldots \leq c *\) card \(P\) using sum-lower-bound-card[OF \(\operatorname{assms}(1)]\).
    finally show ?thesis by (meson discrete nat-mult-less-cancel-disj of-nat-less-imp-less)
qed
lemma \(P_{1}\) - \(B_{1}\)-lower-bound-card:
    assumes bp \(P \operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2}(S \cup L)\) bij-exists \(P_{1}\left(\cup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)\)
    shows card \(\left(P_{1} \cup\right.\) wrap \(\left.B_{1}\right) \leq \operatorname{card} P\)
proof -
    have card \(\left(P_{1} \cup\right.\) wrap \(\left.B_{1}\right) \leq \operatorname{card} P_{1}+\operatorname{card}\left(\right.\) wrap \(\left.B_{1}\right)\)
        using card-Un-le by blast
    also have \(\ldots \leq\) card \(P_{1}+1\) using wrap-card by simp
    also have \(\ldots \leq\) card \(P\) using \(P_{1}\)-lower-bound-card[OF assms].
    finally show ?thesis.
qed
lemma \(P_{2}\) - \(B_{2}\)-lower-bound- \(P_{1}\) :
    assumes inv \(P_{1} P_{2} B_{1} B_{2}(S \cup L) 2 * \operatorname{card} P_{2} \leq \operatorname{card}\left(\bigcup P_{2}\right)\) bij-exists \(P_{1}\)
\(\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)\)
    shows \(2 * \operatorname{card}\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right) \leq \operatorname{card} P_{1}+1\)
proof -
    note invrules \(=\operatorname{inv}_{1} E[\) OF assms(1) \(]\) and bprules \(=b p E[O F \operatorname{invrules}(1)]\)
    have pairwise disjnt \(\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\)
        using bprules(1) pairwise-subset by blast
    moreover have \(B_{2} \notin P_{2}\) using invrules(4) by simp
    ultimately have DISJNT: \(\bigcup P_{2} \cap B_{2}=\{ \}\)
        by (auto, metis (no-types, opaque-lifting) sup-bot.right-neutral Un-insert-right
disjnt-iff \(m k\)-disjoint-insert pairwise-insert wrap-Un)
    have finite \(\left(\bigcup P_{2}\right)\) using \(U\)-Finite bprules(3) by auto
    have finite \(B_{2}\) using bp-bins-finite[OF invrules(1)] wrap-not-empty by blast
    have finite \(P_{2}\) finite (wrap \(B_{2}\) ) using bp-sol-finite[OF invrules(1)] by blast+
    have DISJNT2: \(P_{2} \cap\) wrap \(B_{2}=\{ \}\) unfolding wrap-def using \(\left\langle B_{2} \notin P_{2}\right\rangle\) by
auto
    have card \(\left(\right.\) wrap \(\left.B_{2}\right) \leq\) card \(B_{2}\)
    proof (cases \(\left.\left\langle B_{2}=\{ \}\right\rangle\right)\)
            case False
            then have \(1 \leq\) card \(B_{2}\) by (simp add: leI〈finite \(\left.B_{2}\right\rangle\) )
            then show ?thesis using wrap-card \(\left[\right.\) of \(\left.B_{2}\right]\) by linarith
    qed simp
```

- This part of the proof is based on the proof on page 73 of the article [2].
from $\operatorname{assms}(2)$ have $2 * \operatorname{card} P_{2}+2 * \operatorname{card}\left(\right.$ wrap $\left.B_{2}\right) \leq \operatorname{card}\left(\bigcup P_{2}\right)+\operatorname{card}$ $\left(\right.$ wrap $\left.B_{2}\right)+1$
using wrap-card[of $B_{2}$ ] by linarith
then have 2 $*\left(\operatorname{card} P_{2}+\operatorname{card}\left(\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \leq \operatorname{card}\left(\bigcup P_{2}\right)+\operatorname{card} B_{2}+1$
using $\left\langle\operatorname{card}\left(\right.\right.$ wrap $\left.B_{2}\right) \leq$ card $\left.B_{2}\right\rangle$ by simp

```
    then have 2 \(*\left(\operatorname{card}\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right) \leq \operatorname{card}\left(\bigcup P_{2} \cup B_{2}\right)+1\)
    using card-Un-disjoint[OF <finite \(\left.\left(\bigcup P_{2}\right)\right\rangle\left\langle\right.\) finite \(\left.B_{2}\right\rangle\) DISJNT]
    and card-Un-disjoint \(\left[O F\left\langle\right.\right.\) finite \(\left.P_{2}\right\rangle\left\langle\right.\) finite \(\left(\right.\) wrap \(\left.\left.B_{2}\right)\right\rangle\) DISJNT2] by argo
    then have \(2 *\left(\operatorname{card}\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right) \leq \operatorname{card}\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)+1\)
    by (cases \(\left\langle B_{2}=\{ \}\right\rangle\) ) (auto simp: Un-commute)
then show \(2 *\left(\operatorname{card}\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right) \leq \operatorname{card} P_{1}+1\)
    using assms(3) bij-betw-same-card unfolding bij-exists-def by metis
qed
```

We add $S L S L$ to $i n v_{2}$ to ensure that the $S$ and $L$ sets only contain objects with correct weights.
definition $i n v_{2}::$ ' $a$ set set $\Rightarrow{ }^{\prime} a$ set set $\Rightarrow{ }^{\prime} a$ set $\Rightarrow{ }^{\prime} a$ set $\Rightarrow{ }^{\prime} a$ set $\Rightarrow{ }^{\prime} a$ set $\Rightarrow$ bool where
$i n v_{2} P_{1} P_{2} B_{1} B_{2} S L \longleftrightarrow i n v_{1} P_{1} P_{2} B_{1} B_{2}(S \cup L)-i n v_{1}$ holds for the partial solution
$\wedge\left(L \neq\{ \} \longrightarrow\left(\forall B \in P_{1} \cup\right.\right.$ wrap $\left.\left.B_{1} . B \cap L_{U} \neq\{ \}\right)\right)$ - If there are still large objects left, then every bin of the first partial solution must contain a large object
$\wedge$ bij-exists $P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ - There exists a bijective function between the bins of the first partial solution and the objects of the second one
$\wedge\left(2 * \operatorname{card} P_{2} \leq \operatorname{card}\left(\bigcup P_{2}\right)\right)$ - There are at most twice as many bins in $P_{2}$ as there are objects in $P_{2}$

$$
\wedge S L S L-S \text { and } L \text { are subsets of } S_{U} \text { and } L_{U}
$$

```
lemma \(\mathrm{inv}_{2} E\) :
    assumes \(\operatorname{inv}_{2} P_{1} P_{2} B_{1} B_{2} S L\)
    shows inv \({ }_{1} P_{1} P_{2} B_{1} B_{2}(S \cup L)\)
        and \(L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup\) wrap \(B_{1} . B \cap L_{U} \neq\{ \}\)
        and bij-exists \(P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)\)
        and \(2 *\) card \(P_{2} \leq \operatorname{card}\left(\bigcup P_{2}\right)\)
        and \(S L S L\)
    using assms unfolding inv \(v_{2}\)-def by blast+
lemma \(i n v_{2} I\) :
    assumes \(\operatorname{inv}_{1} P_{1} P_{2} B_{1} B_{2}(S \cup L)\)
        and \(L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup\) wrap \(B_{1} . B \cap L_{U} \neq\{ \}\)
        and bij-exists \(P_{1}\left(\cup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)\)
        and \(2 *\) card \(P_{2} \leq \operatorname{card}\left(\bigcup P_{2}\right)\)
        and \(S L S L\)
    shows \(\mathrm{inv}_{2} P_{1} P_{2} B_{1} B_{2} S L\)
    using assms unfolding inv \(_{2}\)-def by blast
lemma bin-packing-lower-bound-card:
    assumes \(S=\{ \}\) inv \(_{2} P_{1} P_{2} B_{1} B_{2} S L b p P\)
    shows card \(\left(P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(\left.B_{2} \cup\{\{v\} \mid v . v \in S \cup L\}\right) \leq 3 / 2 *\)
card \(P\)
proof (cases \(\langle L=\{ \}\rangle\) )
    note invrules \(=\) inv \(_{2} E[\) OF assms(2) \(]\)
```


## case True

then have card $\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $\left.B_{2} \cup\{\{v\} \mid v . v \in S \cup L\}\right)$ $=\operatorname{card}\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $\left.B_{2}\right)$ using assms(1) by simp
also have $\ldots \leq \operatorname{card}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)+\operatorname{card}\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)$
using card-Un-le[of $\left\langle P_{1} \cup\right.$ wrap $\left.\left.B_{1}\right\rangle\right]$ by (simp add: Un-assoc)
also have $\ldots \leq \operatorname{card} P+\operatorname{card}\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)$
using $P_{1}-B_{1}$-lower-bound-card $[$ OF $\operatorname{assms}(3)$ invrules $(1,3)]$ by simp
also have $\ldots \leq \operatorname{card} P+\operatorname{card} P / 2$
using $P_{2}-B_{2}$-lower-bound- $P_{1}[$ OF invrules $(1,4,3)]$
and $P_{1}$-lower-bound-card $[$ OF assms(3) invrules (1,3)] by linarith
finally show ?thesis by linarith
next
note invrules $=$ inv $_{2} E[O F$ assms(2) $]$
case False
have $\operatorname{card}\left(P_{1} \cup\right.$ wrap $B_{1} \cup P_{2} \cup$ wrap $\left.B_{2} \cup\{\{v\} \mid v . v \in S \cup L\}\right)$ $=\operatorname{card}\left(P_{1} \cup\right.$ wrap $B_{1} \cup\{\{v\} \mid v . v \in L\} \cup P_{2} \cup$ wrap $\left.B_{2}\right)$
using assms(1) by (simp add: Un-commute Un-assoc)
also have $\ldots \leq \operatorname{card}\left(P_{1} \cup\right.$ wrap $\left.B_{1} \cup\{\{v\} \mid v . v \in L\}\right)+\operatorname{card}\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)$
using card-Un-le[of $\left\langle P_{1} \cup\right.$ wrap $\left.\left.B_{1} \cup\{\{v\} \mid v . v \in L\}\right\rangle\right]$ by (simp add: Un-assoc)
also have $\ldots \leq \operatorname{card} P+\operatorname{card}\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)$
using L-bins-lower-bound-card[OF assms(3) invrules(1) invrules(2)[OF False]
invrules(5)] by linarith
also have $\ldots \leq \operatorname{card} P+\operatorname{card} P / 2$
using $P_{2}$ - $B_{2}$-lower-bound- $P_{1}[O F$ invrules $(1,4,3)]$
and $P_{1}$-lower-bound-card $[$ OF $\operatorname{assms}(3) \operatorname{invrules}(1,3)]$ by linarith
finally show?thesis by linarith
qed
definition inv $_{3}::$ ' $a$ set set $\Rightarrow$ ' $a$ set set $\Rightarrow$ 'a set $\Rightarrow{ }^{\prime} a$ set $\Rightarrow{ }^{\prime} a$ set $\Rightarrow$ ' $a$ set $\Rightarrow$ bool where
$\operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} S L \longleftrightarrow \operatorname{inv}_{2} P_{1} P_{2} B_{1} B_{2} S L \wedge B_{2} \subseteq S_{U}$
lemma $i n v_{3} E$ :
assumes $\operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} S L$
shows inv $P_{1} P_{2} B_{1} B_{2} S L$ and $B_{2} \subseteq S_{U}$
using assms unfolding inv ind $_{3}$ def by blast+
lemma $i n v_{3} I$ :
assumes $i n v_{2} P_{1} P_{2} B_{1} B_{2} S L$ and $B_{2} \subseteq S_{U}$
shows inv ${ }_{3} P_{1} P_{2} B_{1} B_{2} S L$
using assms unfolding $i n v_{3}$-def by blast
lemma loop-init:
inv $_{3}\{ \}\{ \}\{ \}\{ \} S_{U} L_{U}$
proof -
have $S_{U} \cup L_{U}=U$ by auto
then have $*: \operatorname{inv}_{1}\{ \}\{ \}\{ \}\{ \}\left(S_{U} \cup L_{U}\right)$
unfolding bp-def partition-on-def pairwise-def wrap-def inv $v_{1}$-def
using weight by auto

```
    have bij-exists {} (U ({} \cup wrap {}))
    using bij-betwI' unfolding bij-exists-def by fastforce
```



```
    from inv I}I[OF this] show ?thesis by blas
qed
lemma loop-stepA:
    assumes inv 3}\mp@subsup{P}{1}{}\mp@subsup{P}{2}{}\mp@subsup{B}{1}{}\mp@subsup{B}{2}{}SL\mp@subsup{B}{1}{}={}L={}u\in
    shows inv 3 P P P P {u} B (S-{u})L
proof -
    note invrules = inv 2 E[OF inv E E(1)[OF assms(1)]]
    have WEIGHT:W B 位 + wu\leqc using invrules(5) assms(2,4) by fastforce
    from assms(4) have }u\inS\cupL\mathrm{ by blast
    from inv -stepA[OF invrules(1) this WEIGHT] assms(2,3) have 1: inv ( P P P P P
{u} B2 (S-{u}\cupL) by simp
    have 2: }L\not={}\Longrightarrow\forallB\in\mp@subsup{P}{1}{}\cup\mathrm{ wrap {u}. B }\cap\mp@subsup{L}{U}{}\not={}\mathrm{ using assms(3) by blast
    from inv }I[OF 1 2] invrules have inv 2 P1 P P {u} B2 (S-{u})L by blas
    from inv }I[OF this] show ?thesis using inv E(2)[OF assms(1)]
qed
lemma loop-stepB:
    assumes inv ( P P P P B B B S S L B = ={}u\inL
    shows inv 3 P P P P { {u} B S S (L-{u})
proof -
    note invrules = inv 2 E[OF inv }E\mathrm{ (1)[OF assms(1)]]
    have WEIGHT:W B 的 + wu\leqc using weight invrules(5) assms(2,3) by
fastforce
- This observation follows from the fact that the \(S\) and \(L\) sets have to be disjoint from each other, and allows us to reuse our proofs of the preservation of \(i n v_{1}\) by simply replacing \(V\) with \(S \cup L\)
have \(*: S \cup L-\{u\}=S \cup(L-\{u\})\) using invrules(5) assms(3) by force from \(\operatorname{assms}(3)\) have \(u \in S \cup L\) by blast
from inv \(v_{1}\)-step \(A\left[O F\right.\) invrules(1) this WEIGHT] assms(2) * have 1: inv \({ }_{1} P_{1} P_{2}\) \(\{u\} B_{2}(S \cup(L-\{u\}))\) by \(\operatorname{simp}\)
have \(\forall B \in P_{1} . B \cap L_{U} \neq\{ \}\{u\} \cap L_{U} \neq\{ \}\) using \(\operatorname{assms}(3)\) invrules \((2,5)\) by blast+
then have 2: \(L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup\) wrap \(\{u\} . B \cap L_{U} \neq\{ \}\)
using assms(3) by (metis (full-types) Un-iff empty-iff insert-iff wrap-not-empty)
from \(i n v_{2} I\left[O F 1\right.\) 2] invrules have \(i n v_{2} P_{1} P_{2}\{u\} B_{2} S(L-\{u\})\) by blast
from inv \({ }_{3} I[O F\) this \(]\) show ?thesis using inv \(_{3} E(2)[O F \operatorname{assms}(1)]\).
qed
lemma loop-step \(C\) :
assumes \(\operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} S L B_{1} \neq\{ \} u \in S W B_{1}+w(u) \leq c\)
shows \(\operatorname{inv}_{3} P_{1} P_{2}\left(B_{1} \cup\{u\}\right) B_{2}(S-\{u\}) L\)
proof -
note invrules \(=\operatorname{inv}_{2} E\left[O F\right.\) inv \(\left._{3} E(1)[O F \operatorname{assms}(1)]\right]\)
```

- Same approach, but removing $\{u\}$ from $S$ instead of $L$
have $*: S \cup L-\{u\}=(S-\{u\}) \cup L$ using invrules(5) assms(3) by force
from $\operatorname{assms}(3)$ have $u \in S \cup L$ by blast
from $\operatorname{inv}_{1}$-step $A\left[O F \operatorname{invrules}(1)\right.$ this assms(4)] * have 1: inv $P_{1} P_{2}\left(B_{1} \cup\right.$ $\{u\}) B_{2}(S-\{u\} \cup L)$ by simp
have $L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup$ wrap $B_{1} . B \cap L_{U} \neq\{ \}$ using invrules(2) by blast
then have 2: $L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup$ wrap $\left(B_{1} \cup\{u\}\right) . B \cap L_{U} \neq\{ \}$
by (smt Int-insert-left Un-empty-right Un-iff Un-insert-right assms(2) in-sert-not-empty singletonD singletonI wrap-def)
from $\mathrm{inv}_{2} I\left[O F 1\right.$ 2] invrules have $i n v_{2} P_{1} P_{2}\left(B_{1} \cup\{u\}\right) B_{2}(S-\{u\}) L$ by blast
from $i n v_{3} I[$ OF this $]$ show ?thesis using inv $_{3} E($ 2) $[$ OF assms(1)].
qed
lemma loop-stepD:
assumes $i n v_{3} P_{1} P_{2} B_{1} B_{2} S L B_{1} \neq\{ \} u \in S W B_{1}+w(u)>c W B_{2}+$ $w(u) \leq c$
shows $\operatorname{inv}_{3}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right) P_{2}\{ \}\left(B_{2} \cup\{u\}\right)(S-\{u\}) L$
proof -
note invrules $=$ inv $_{2} E\left[\right.$ OF inv in $_{3} E(1)[$ OF assms(1) $\left.]\right]$
have $*: S \cup L-\{u\}=(S-\{u\}) \cup L$ using invrules(5) assms(3) by force
from $\operatorname{assms}(3)$ have $u \in S \cup L$ by blast
from inv $\mathrm{in}_{1}$ step $B\left[O F\right.$ invrules (1) this assms(5)] * have 1: inv ${ }_{1}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)$ $P_{2}\{ \}\left(B_{2} \cup\{u\}\right)(S-\{u\} \cup L)$ by simp
have 2: $L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup$ wrap $B_{1} \cup$ wrap $\left\} . B \cap L_{U} \neq\{ \}\right.$ using invrules(2) unfolding wrap-empty by blast
from invrules(3) obtain $f$ where $f$-def: bij-betw $f P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ $\forall B \in P_{1} . c<W B+w(f B)$ unfolding bij-exists-def by blast
have $B_{1} \notin P_{1}$ using inv $E(3)$ [OF invrules(1)] by blast
have $u \notin\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ using $\operatorname{inv}_{1} E(2)[O F \operatorname{invrules}(1)] \operatorname{assms}(3)$ by blast
then have $\left(\bigcup\left(P_{2} \cup \operatorname{wrap}\left(B_{2} \cup\{u\}\right)\right)\right)=\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2} \cup\{\{u\}\}\right)\right)$
by (metis Sup-empty Un-assoc Union-Un-distrib ccpo-Sup-singleton wrap-empty wrap-not-empty)
also have $\ldots=\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \cup\{u\}$ by simp
finally have $U N:\left(\bigcup\left(P_{2} \cup \operatorname{wrap}\left(B_{2} \cup\{u\}\right)\right)\right)=\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right) \cup\{u\}$.
have wrap $B_{1}=\left\{B_{1}\right\}$ using wrap-not-empty[of $\left.B_{1}\right]$ assms(2) by simp
let ?f $=f\left(B_{1}:=u\right)$
have BIJ: bij-betw ?f $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\cup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.\left(B_{2} \cup\{u\}\right)\right)\right)$
unfolding wrap-empty «wrap $\left.B_{1}=\left\{B_{1}\right\}\right\rangle U N$ using $f$-def(1) $\left\langle B_{1} \notin P_{1}\right\rangle\langle u \notin$
$\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ >
by (metis (no-types, lifting) bij-betw-cong fun-upd-other fun-upd-same notIn-Un-bij-betw3)
have $c<W B_{1}+w\left(\right.$ ?f $\left.B_{1}\right)$ using $\operatorname{assms}(4)$ by simp
then have $\left(\forall B \in P_{1} \cup\right.$ wrap $B_{1} . c<W B+w($ ?f $\left.B)\right)$
unfolding $\left\langle\right.$ wrap $B_{1}=\left\{B_{1}\right\}$ 〉 using $f$ - $\operatorname{def}($ (2) by simp
with $B I J$ have bij-betw ?f $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\cup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.\left(B_{2} \cup\{u\}\right)\right)\right)$
$\wedge\left(\forall B \in P_{1} \cup\right.$ wrap $B_{1} . c<W B+w($ ?f $\left.B)\right)$ by blast
then have 3: bij-exists $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\bigcup\left(P_{2} \cup \operatorname{wrap}\left(B_{2} \cup\{u\}\right)\right)\right)$
unfolding bij-exists-def by blast
from $\operatorname{inv}_{2} I[O F 123]$ have $\operatorname{inv}_{2}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right) P_{2}\{ \}\left(B_{2} \cup\{u\}\right)(S-\{u\})$ $L$ using invrules $(4,5)$ by blast
from $\mathrm{inv}_{3} I[O F$ this $]$ show ?thesis using $\operatorname{inv}_{3} E(2)[O F \operatorname{assms}(1)] \operatorname{assms}(3)$ invrules(5) by blast
qed
lemma $B_{2}$-at-least-two-objects:
assumes $\operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} S L u \in S W B_{2}+w(u)>c$
shows $2 \leq$ card $B_{2}$
proof (rule ccontr, simp add: not-le)
have FINITE: finite $B_{2}$ using $\operatorname{inv}_{1} E(1)\left[O F \operatorname{inv}_{2} E(1)\left[O F \operatorname{inv}_{3} E(1)[O F \operatorname{assms}(1)]\right]\right]$ by (metis (no-types, lifting) Finite-Set.finite.simps U-Finite Union-Un-distrib
$b p E$ (3) ccpo-Sup-singleton finite-Un wrap-not-empty)
assume card $B_{2}<2$
then consider ( 0 ) card $B_{2}=0 \mid$ (1) card $B_{2}=1$ by linarith
then show False proof cases
case 0 then have $B_{2}=\{ \}$ using FINITE by simp
then show ?thesis using $\operatorname{assms}(2,3) \operatorname{inv}_{2} E(5)\left[O F \operatorname{inv}_{3} E(1)[O F \operatorname{assms}(1)]\right]$
by force
next
case 1 then obtain $v$ where $B_{2}=\{v\}$
using card-1-singletonE by auto
with $\operatorname{inv}_{3} E$ (2)[OF assms(1)] have $2 * w v \leq c$ using $\operatorname{inv}_{2} E(5)\left[O F \operatorname{inv} v_{3} E(1)[O F\right.$ $\operatorname{assms}(1)]]$ by $\operatorname{simp}$
moreover from $\left\langle B_{2}=\{v\}\right\rangle$ have $W B_{2}=w v$ by simp
ultimately show ?thesis using $\operatorname{assms}(2,3) \mathrm{inv}_{2} E(5)\left[O F \operatorname{inv}_{3} E(1)[\right.$ OF $\left.\operatorname{assms}(1)]\right]$ by force
qed
qed
lemma loop-stepE:
assumes $\operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} S L B_{1} \neq\{ \} u \in S W B_{1}+w(u)>c W B_{2}+$ $w(u)>c$
shows $\operatorname{inv}_{3}\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)\{ \}\{u\}(S-\{u\}) L$
proof -
note invrules $=$ inv $_{2} E\left[O F\right.$ inv $_{3} E(1)[$ OF assms(1) $\left.]\right]$
have $*: S \cup L-\{u\}=(S-\{u\}) \cup L$ using invrules(5) assms(3) by force
from $\operatorname{assms}(3)$ have $u \in S \cup L$ by blast
from inv $v_{1}-$ step $C[$ OF invrules (1) this $] *$ have 1: inv $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(P_{2} \cup\right.$ wrap $\left.B_{2}\right)\}\{u\}(S-\{u\} \cup L)$ by simp
have 2: $L \neq\{ \} \Longrightarrow \forall B \in P_{1} \cup$ wrap $B_{1} \cup$ wrap $\left\} . B \cap L_{U} \neq\{ \}\right.$
using invrules(2) unfolding wrap-empty by blast
from invrules(3) obtain $f$ where $f$-def: bij-betw $f P_{1}\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ $\forall B \in P_{1} . c<W B+w(f B)$ unfolding bij-exists-def by blast
have $B_{1} \notin P_{1}$ using $\operatorname{inv}_{1} E(3)[$ OF invrules(1)] by blast
have $u \notin\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ using $\operatorname{inv}_{1} E($ 2) $[O F$ invrules(1)] assms(3) by blast
have $\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $B_{2} \cup$ wrap $\left.\left.\{u\}\right)\right)=\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2} \cup\{\{u\}\}\right)\right)$ unfolding wrap-def by simp

```
    also have \(\ldots=\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right) \cup\{u\}\) by simp
```

    finally have \(U N:\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(B_{2} \cup\) wrap \(\left.\left.\{u\}\right)\right)=\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right) \cup\)
    $\{u\}$.
have wrap $B_{1}=\left\{B_{1}\right\}$ using wrap-not-empty[of $\left.B_{1}\right]$ assms(2) by simp
let ?f $=f\left(B_{1}:=u\right)$
have BIJ: bij-betw ?f $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $B_{2} \cup$ wrap $\left.\left.\{u\}\right)\right)$
unfolding wrap-empty «wrap $\left.B_{1}=\left\{B_{1}\right\}\right\rangle U N$ using $f$ - $\operatorname{def}(1)\left\langle B_{1} \notin P_{1}\right\rangle\langle u \notin$
$\left(\cup\left(P_{2} \cup\right.\right.$ wrap $\left.\left.B_{2}\right)\right)$ >
by (metis (no-types, lifting) bij-betw-cong fun-upd-other fun-upd-same notIn-Un-bij-betw3)
have $c<W B_{1}+w\left(\right.$ ?f $\left.B_{1}\right)$ using $\operatorname{assms}(4)$ by simp
then have $\left(\forall B \in P_{1} \cup\right.$ wrap $\left.B_{1} . c<W B+w(? f B)\right)$
unfolding $\left\langle\right.$ wrap $\left.B_{1}=\left\{B_{1}\right\}\right\rangle$ using $f$ - $\operatorname{def}($ (2) by simp
with BIJ have bij-betw? $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $B_{2} \cup$ wrap $\left.\left.\{u\}\right)\right)$
$\wedge\left(\forall B \in P_{1} \cup\right.$ wrap $B_{1} . c<W B+w($ ?f $\left.B)\right)$ by blast
then have 3: bij-exists $\left(P_{1} \cup\right.$ wrap $\left.B_{1}\right)\left(\bigcup\left(P_{2} \cup\right.\right.$ wrap $B_{2} \cup$ wrap $\left.\left.\{u\}\right)\right)$
unfolding bij-exists-def by blast

```
have 4:2* card \(\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right) \leq \operatorname{card}\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)\)
proof -
    note bprules \(=b p E\left[\right.\) OF inv \({ }_{1} E(1)[\) OF invrules (1) \(\left.]\right]\)
    have pairwise disjnt \(\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\)
        using bprules(1) pairwise-subset by blast
    moreover have \(B_{2} \notin P_{2}\) using \(\operatorname{inv}_{1} E(4)[\) OF invrules(1)] by simp
    ultimately have DISJNT: \(\bigcup P_{2} \cap B_{2}=\{ \}\)
        by (auto, metis (no-types, opaque-lifting) sup-bot.right-neutral Un-insert-right
disjnt-iff mk-disjoint-insert pairwise-insert wrap-Un)
    have finite \(\left(\bigcup P_{2}\right)\) using \(U\)-Finite bprules(3) by auto
        have finite \(B_{2}\) using inv \(_{1} E(1)[O F\) invrules (1)] bp-bins-finite wrap-not-empty
by blast
    have \(2 * \operatorname{card}\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right) \leq 2 *\left(\operatorname{card} P_{2}+\operatorname{card}\left(\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)\)
        using card-Un-le[of \(P_{2}\) 〈wrap \(B_{2}\) 〉] by simp
    also have \(\ldots \leq 2 *\) card \(P_{2}+2\) using wrap-card by auto
    also have \(\ldots \leq \operatorname{card}\left(\bigcup P_{2}\right)+2\) using invrules(4) by simp
        also have \(\ldots \leq \operatorname{card}\left(\bigcup P_{2}\right)+\) card \(B_{2}\) using \(B_{2}\)-at-least-two-objects[OF
\(\operatorname{assms}(1,3,5)]\) by \(\operatorname{simp}\)
    also have \(\ldots=\operatorname{card}\left(\bigcup\left(P_{2} \cup\left\{B_{2}\right\}\right)\right)\) using DISJNT card-Un-disjoint[OF
\(\left\langle\right.\) finite \(\left.\left(\bigcup P_{2}\right)\right\rangle\left\langle\right.\) finite \(\left.\left.B_{2}\right\rangle\right]\) by (simp add: Un-commute)
    also have \(\ldots=\) card \(\left(\bigcup\left(P_{2} \cup\right.\right.\) wrap \(\left.\left.B_{2}\right)\right)\) by (cases \(\left.\left\langle B_{2}=\{ \}\right\rangle\right)\) auto
    finally show ?thesis .
    qed
    from \(\operatorname{inv}_{2} I[O F 1234]\) have \(\operatorname{inv}_{2}\left(P_{1} \cup\right.\) wrap \(\left.B_{1}\right)\left(P_{2} \cup\right.\) wrap \(\left.B_{2}\right)\{ \}\{u\}(S\)
- \(\{u\}) L\)
    using invrules(5) by blast
```

from inv $_{3} I[$ OF this] show ?thesis using assms(3) invrules(5) by blast qed

The bin packing algorithm as it is proposed on page 78 of the article [2]. $P$ will not only be a correct solution of the bin packing problem, but the amount of bins will be a lower bound for $3 / 2$ of the amount of bins of any correct solution $Q$, and thus guarantee an approximation factor of $3 / 2$ for the optimum.

```
lemma bp-approx:
\(V A R S P P_{1} P_{2} B_{1} B_{2} V S L u\)
    \{True\}
    \(S:=\{ \} ; L:=\{ \} ; V:=U\);
    WHILE \(V \neq\{ \} \operatorname{INV}\{V \subseteq U \wedge S=\{u \in U-V . w(u) \leq c / 2\} \wedge L=\{u \in\)
\(U-V . c / 2<w(u)\}\} D O\)
    \(u:=(S O M E \quad u . u \in V)\);
    IF \(w(u) \leq c / 2\)
    THEN \(S:=S \cup\{u\}\)
    ELSE \(L:=L \cup\{u\} F I\);
        \(V:=V-\{u\}\)
    \(O D\);
    \(P_{1}:=\{ \} ; P_{2}:=\{ \} ; B_{1}:=\{ \} ; B_{2}:=\{ \} ;\)
    WHILE \(S \neq\{ \}\) INV \(\left\{\right.\) inv \(\left._{3} P_{1} P_{2} B_{1} B_{2} S L\right\} D O\)
        IF \(B_{1} \neq\{ \}\)
        THEN \(u:=(S O M E\) u. \(u \in S) ; S:=S-\{u\}\)
    ELSE IF \(L \neq\{ \}\)
                THEN \(u:=(\) SOME \(u . u \in L) ; L:=L-\{u\}\)
                ELSE \(u:=(S O M E u . u \in S) ; S:=S-\{u\}\) FI FI;
    IF \(W\left(B_{1}\right)+w(u) \leq c\)
    THEN \(B_{1}:=B_{1} \cup\{u\}\)
    ELSE IF \(W\left(B_{2}\right)+w(u) \leq c\)
                THEN \(B_{2}:=B_{2} \cup\{u\}\)
                \(E L S E P_{2}:=P_{2} \cup\) wrap \(B_{2} ; B_{2}:=\{u\} F I ;\)
                \(P_{1}:=P_{1} \cup\) wrap \(B_{1} ; B_{1}:=\{ \} F I\)
    \(O D\);
    \(P:=P_{1} \cup\) wrap \(B_{1} \cup P_{2} \cup\) wrap \(B_{2} ; V:=L ;\)
    WHILE \(V \neq\{ \}\)
    \(\operatorname{INV}\left\{S=\{ \} \wedge \operatorname{inv}_{3} P_{1} P_{2} B_{1} B_{2} S L \wedge V \subseteq L \wedge P=P_{1} \cup\right.\) wrap \(B_{1} \cup P_{2} \cup\)
wrap \(\left.B_{2} \cup\{\{v\} \mid v . v \in L-V\}\right\} D O\)
    \(u:=(S O M E u . u \in V) ; P:=P \cup\{\{u\}\} ; V:=V-\{u\}\)
    \(O D\)
    \(\{b p P \wedge(\forall Q . b p Q \longrightarrow \operatorname{card} P \leq 3 / 2 *\) card \(Q)\}\)
proof (vcg, goal-cases)
    case ( \(1 P P_{1} P_{2} B_{1} B_{2} V S L u\) )
    then show ?case by blast
next
    case (2 P \(P_{1} P_{2} B_{1} B_{2}\) V SLu)
    then show? case by (auto simp: some-in-eq)
next
```



```
    then show ?case using loop-init by force
next
    case (4 P P Pr P
    then have INV:inv i Pr P
    let ?s = SOME u.u\inS
    let ?l = SOME u.u\inL
    note SL-def = inv 2E(5)[OF inv 3E(1)[OF INV]]
    have LIN: L\not={}\Longrightarrow?l }\inL\mathrm{ using some-in-eq by metis
    then have LWEIGHT:L\not={}\Longrightarroww?l 
    from 4 have S\not={} ..
    then have IN: ?s \inS using some-in-eq by metis
    then have w?s\leqc using SL-def by auto
    then show ?case
    using LWEIGHT loop-stepA[OF INV - - IN] loop-stepB[OF INV - LIN]
loop-stepC[OF INV - IN]
    and loop-stepD[OF INV -IN] loop-stepE[OF INV - IN] by (cases <B
cases }\langleL={}>) aut
next
```



```
    then show ?case by blast
next
```



```
    then have *:(SOME u.u\inV)\inV (SOME u.u\inV)\inL by (auto simp add:
some-in-eq)
    then have }\mp@subsup{P}{1}{}\cup\mathrm{ wrap B}\mp@subsup{B}{1}{}\cup\mp@subsup{P}{2}{}\cup\mathrm{ wrap B}\mp@subsup{B}{2}{}\cup{{v}|v.v\inL-(V - {SOME u
u\inV})}
                    = P
\inV}}
            by blast
    with 6 * show ?case by blast
next
```



```
    then have *: inv2 P
        using inv }\mp@subsup{v}{3}{}E(1)\mathrm{ by blast
```



```
    have bp P by fastforce
    with bin-packing-lower-bound-card[OF - *] 7
    show ?case by fastforce
qed
end
end
```


## 6 Center Selection

theory Center-Selection
imports Complex-Main HOL-Hoare.Hoare-Logic

## begin

The Center Selection (or metric k-center) problem. Given a set of sites $S$ in a metric space, find a subset $C \subseteq S$ that minimizes the maximal distance from any $s \in S$ to some $c \in C$. This theory presents a verified 2-approximation algorithm. It is based on Section 11.2 in the book by Kleinberg and Tardos [4]. In contrast to the proof in the book, our proof is a standard invariant proof.

```
locale Center-Selection =
    fixes \(S::\left({ }^{\prime} a::\right.\) metric-space) set
        and \(k::\) nat
    assumes finite-sites: finite \(S\)
    and non-empty-sites: \(S \neq\{ \}\)
and non-zero- \(k: k>0\)
begin
```

definition distance :: ('a::metric-space) set $\Rightarrow$ ('a::metric-space) $\Rightarrow$ real where
distance $C s=\operatorname{Min}\left(\right.$ dist $\left.s{ }^{\prime} C\right)$
definition radius :: (' $a$ :: metric-space) set $\Rightarrow$ real where
radius $C=\operatorname{Max}$ (distance $C$ ' $S$ )
lemma distance-mono:
assumes $C_{1} \subseteq C_{2}$ and $C_{1} \neq\{ \}$ and finite $C_{2}$
shows distance $C_{1} s \geq$ distance $C_{2} s$
by (simp add: Min.subset-imp assms distance-def image-mono)
lemma finite-distances: finite (distance $C$ ' $S$ )
using finite-sites by simp
lemma non-empty-distances: distance $C$ ' $S \neq\{ \}$
using non-empty-sites by simp
lemma radius-contained: radius $C \in$ distance $C$ ' $S$
using finite-distances non-empty-distances Max-in radius-def by simp
lemma radius-def2: $\exists s \in S$. distance $C s=$ radius $C$
using radius-contained image-iff by metis
lemma dist-lemmas-aux:
assumes finite $C$
and $C \neq\{ \}$
shows finite (dist s' $C$ )
and finite (dist s'C) $\Longrightarrow$ distance $C s \in$ dist s ' $C$
and distance $C s \in$ dist $s{ }^{\prime} C \Longrightarrow \exists c \in C$. dist $s c=$ distance $C s$
and $\exists c \in C$. dist $s c=$ distance $C s \Longrightarrow$ distance $C s \geq 0$
proof
show finite $C$ using assms(1) by simp
next

```
    assume finite (dist s'C)
    then show distance C s\in dist s 'C using distance-def eq-Min-iff assms(2) by
blast
next
    assume distance C s \in dist s ' C
    then show }\existsc\inC\mathrm{ . dist s c = distance C s by auto
next
    assume \existsc\inC. dist s c= distance C s
    then show distance Cs\geq0 by (metis zero-le-dist)
qed
lemma dist-lemmas:
    assumes finite C
        and C\not={}
    shows finite (dist s ' C)
        and distance Cs\indist s'C
        and \existsc\inC. dist s c = distance Cs
        and distance Cs\geq0
    using dist-lemmas-aux assms by auto
lemma radius-max-prop: (\foralls\inS.distance C s \leqr)\Longrightarrow(radius C \leqr)
    by (metis image-iff radius-contained)
lemma dist-ins:
assumes }\forall\mp@subsup{c}{1}{}\inC.\forall\mp@subsup{c}{2}{}\inC.\mp@subsup{c}{1}{}\not=\mp@subsup{c}{2}{}\longrightarrowx<dist \mp@subsup{c}{1}{}\mp@subsup{c}{2}{
and distance Cs>x
and finite C
and C}C\not={
shows }\forall\mp@subsup{c}{1}{}\in(C\cup{s}).\forall\mp@subsup{c}{2}{}\in(C\cup{s}).c\mp@subsup{c}{1}{}\not=\mp@subsup{c}{2}{}\longrightarrowx<dist c c c c c <
proof (rule+)
    fix }\mp@subsup{c}{1}{}\mp@subsup{c}{2}{
    assume local-assms: c}\mp@subsup{c}{1}{}\inC\cup{s} \mp@subsup{c}{2}{}\inC\cup{s} c c \not= c c <
    then have c}\mp@subsup{c}{1}{}\inC\wedge\mp@subsup{c}{2}{}\inC\vee\mp@subsup{c}{1}{}\inC\wedge\mp@subsup{c}{2}{}\in{s}\vee\mp@subsup{c}{2}{}\inC\wedge\mp@subsup{c}{1}{}\in{s}\vee\mp@subsup{c}{1}{}
{s}\wedge c}\mp@subsup{2}{2}{}\in{s}\mathrm{ by auto
    then show }x<dist c) c1 c
    proof (elim disjE)
        assume c}\mp@subsup{c}{1}{}\inC\wedge\mp@subsup{c}{2}{}\in
        then show ?thesis using assms(1) local-assms(3) by simp
    next
    assume case-assm: c}\mp@subsup{c}{1}{}\inC\wedge\mp@subsup{c}{2}{}\in{s
    have }x<\mathrm{ distance C c}\mp@subsup{c}{2}{}\mathrm{ using assms(2) case-assm by simp
    also have ... \leq dist c}\mp@subsup{c}{2}{}\mp@subsup{c}{1}{
        using Min.coboundedI distance-def assms(3,4) dist-lemmas(1, 2) case-assm
by simp
    also have ... = dist c}\mp@subsup{c}{1}{}\mp@subsup{c}{2}{}\mathrm{ using dist-commute by metis
    finally show ?thesis .
next
    assume case-assm: c. c }\inC\wedge\mp@subsup{c}{1}{}\in{s
    have x< distance C c c using assms(2) case-assm by simp
```

```
    also have ...\leq dist c}\mp@subsup{c}{1}{}\mp@subsup{c}{2}{
    using Min.coboundedI distance-def assms(3,4) dist-lemmas(1, 2) case-assm
by simp
    finally show ?thesis.
    next
    assume c}\mp@subsup{c}{1}{}\in{s}\wedge\mp@subsup{c}{2}{}\in{s
    then have False using local-assms by simp
    then show ?thesis by simp
    qed
qed
```


### 6.1 A Preliminary Algorithm and Proof

This subsection verifies an auxiliary algorithm by Kleinberg and Tardos. Our proof of the main algorithm does not does not rely on this auxiliary algorithm at all but we do reuse part off its invariant proof later on.
definition inv :: ('a :: metric-space) set $\Rightarrow$ ('a :: metric-space set) $\Rightarrow$ real $\Rightarrow$ bool where
$\operatorname{inv} S^{\prime} C r=$

$$
\begin{aligned}
& \left(\left(\forall s \in\left(S-S^{\prime}\right) . \text { distance } C s \leq 2 * r\right) \wedge S^{\prime} \subseteq S \wedge C \subseteq S \wedge\right. \\
& \left(\forall c \in C . \forall s \in S^{\prime} \cdot S^{\prime} \neq\{ \} \longrightarrow \text { dist cs>2*r)^(S} S^{\prime}=S \vee C \neq\{ \}\right) \wedge \\
& \left.\left(\forall c_{1} \in C . \forall c_{2} \in C . c_{1} \neq c_{2} \longrightarrow \text { dist } c_{1} c_{2}>2 * r\right)\right)
\end{aligned}
$$

lemma inv-init: inv $S\} r$
unfolding inv-def non-empty-sites by simp
lemma inv-step:
assumes $S^{\prime} \neq\{ \}$
and $I H$ : inv $S^{\prime} C r$
defines $[s i m p]: s \equiv\left(S O M E\right.$ s. $\left.s \in S^{\prime}\right)$
shows inv $\left(S^{\prime}-\left\{s^{\prime} . s^{\prime} \in S^{\prime} \wedge\right.\right.$ dist $\left.\left.s s^{\prime} \leq 2 * r\right\}\right)(C \cup\{s\}) r$
proof -
have $s$-def: $s \in S^{\prime}$ using assms(1) some-in-eq by auto
have finite $(C \cup\{s\})$ using $I H$ finite-subset[OF - finite-sites] by (simp add: inv-def)

## moreover

have $\left(\forall s^{\prime} \in\left(S-\left(S^{\prime}-\left\{s^{\prime} . s^{\prime} \in S^{\prime} \wedge\right.\right.\right.\right.$ dist $\left.\left.\left.s s^{\prime} \leq 2 * r\right\}\right)\right)$. distance $(C \cup\{s\})$ $\left.s^{\prime} \leq 2 * r\right)$
proof
fix $s^{\prime \prime}$
assume $s^{\prime \prime} \in S-\left(S^{\prime}-\left\{s^{\prime} . s^{\prime} \in S^{\prime} \wedge\right.\right.$ dist $\left.\left.s s^{\prime} \leq 2 * r\right\}\right)$
then have $s^{\prime \prime} \in S-S^{\prime} \vee s^{\prime \prime} \in\left\{s^{\prime} . s^{\prime} \in S^{\prime} \wedge\right.$ dist $\left.s s^{\prime} \leq 2 * r\right\}$ by simp
then show distance $(C \cup\{s\}) s^{\prime \prime} \leq 2 * r$
proof (elim disjE)
assume local-assm: $s^{\prime \prime} \in S-S^{\prime}$
have $S^{\prime}=S \vee C \neq\{ \}$ using $I H$ by (simp add: inv-def)
then show ?thesis

```
    proof (elim disjE)
            assume S'=S
            then have s}\mp@subsup{s}{}{\prime\prime}\in{}\mathrm{ using local-assm by simp
            then show ?thesis by simp
        next
            assume C-not-empty: C}\not={
            have finite C using IH finite-subset[OF - finite-sites] by (simp add: inv-def)
            then have distance ( }C\cup{s})\mp@subsup{s}{}{\prime\prime}\leq\mathrm{ distance C s''
            using distance-mono C-not-empty by (meson Un-upper1 calculation)
            also have ... \leq2*r using IH local-assm inv-def by simp
            finally show ?thesis.
        qed
    next
    assume local-assm: s"}\mp@subsup{s}{}{\prime\prime}\in{\mp@subsup{s}{}{\prime}.\mp@subsup{s}{}{\prime}\in\mp@subsup{S}{}{\prime}\wedge\mathrm{ dist s s}\mp@subsup{s}{}{\prime}\leq2*r
    then have distance (C\cup{s}) s'\prime}\leq\mathrm{ dist }\mp@subsup{s}{}{\prime\prime}
        using Min.coboundedI distance-def dist-lemmas calculation by auto
        also have ...\leq2 *r using local-assm by (smt dist-self dist-triangle2
mem-Collect-eq)
            finally show ?thesis .
    qed
qed
moreover
have}\mp@subsup{S}{}{\prime}-{\mp@subsup{s}{}{\prime}.\mp@subsup{s}{}{\prime}\in\mp@subsup{S}{}{\prime}\wedge\mathrm{ dist s s}\mp@subsup{s}{}{\prime}\leq2*r}\subseteqS using IH by (auto simp: inv-def
moreover
{
    have s}\inS\mathrm{ using IH inv-def s-def by auto
    then have C}\cup{s}\subseteqS\mathrm{ using IH by (simp add: inv-def)
}
moreover
```

```
have \(\left(\forall c \in C \cup\{s\} . \forall c_{2} \in C \cup\{s\} . c \neq c_{2} \longrightarrow 2 * r<\operatorname{dist} c c_{2}\right)\)
```

have $\left(\forall c \in C \cup\{s\} . \forall c_{2} \in C \cup\{s\} . c \neq c_{2} \longrightarrow 2 * r<\operatorname{dist} c c_{2}\right)$
proof (rule + )
proof (rule + )
fix $c_{1} c_{2}$
fix $c_{1} c_{2}$
assume local-assms: $c_{1} \in C \cup\{s\} c_{2} \in C \cup\{s\} c_{1} \neq c_{2}$
assume local-assms: $c_{1} \in C \cup\{s\} c_{2} \in C \cup\{s\} c_{1} \neq c_{2}$
then have $\left(c_{1} \in C \wedge c_{2} \in C\right) \vee\left(c_{1}=s \wedge c_{2} \in C\right) \vee\left(c_{1} \in C \wedge c_{2}=s\right) \vee$
then have $\left(c_{1} \in C \wedge c_{2} \in C\right) \vee\left(c_{1}=s \wedge c_{2} \in C\right) \vee\left(c_{1} \in C \wedge c_{2}=s\right) \vee$
$\left(c_{1}=s \wedge c_{2}=s\right)$
$\left(c_{1}=s \wedge c_{2}=s\right)$
using assms by auto
using assms by auto
then show $2 * r<\operatorname{dist} c_{1} c_{2}$
then show $2 * r<\operatorname{dist} c_{1} c_{2}$
proof (elim disjE)
proof (elim disjE)
assume $c_{1} \in C \wedge c_{2} \in C$
assume $c_{1} \in C \wedge c_{2} \in C$
then show $2 * r<d i s t c_{1} c_{2}$ using IH inv-def local-assms by simp
then show $2 * r<d i s t c_{1} c_{2}$ using IH inv-def local-assms by simp
next
next
assume case-assm: $c_{1}=s \wedge c_{2} \in C$
assume case-assm: $c_{1}=s \wedge c_{2} \in C$
have $\left(\forall c \in C . \forall s \in S^{\prime} . S^{\prime} \neq\{ \} \longrightarrow 2 * r<\right.$ dist $\left.c s\right)$ using IH inv-def by
have $\left(\forall c \in C . \forall s \in S^{\prime} . S^{\prime} \neq\{ \} \longrightarrow 2 * r<\right.$ dist $\left.c s\right)$ using IH inv-def by
simp
simp
then show?thesis by (smt case-assm s-def assms(1) dist-self dist-triangle3

```
    then show?thesis by (smt case-assm s-def assms(1) dist-self dist-triangle3
```

```
singletonD)
    next
        assume case-assm: c}\mp@subsup{c}{1}{}\inC\wedge\mp@subsup{c}{2}{}=
        have ( }\forallc\inC.\foralls\in\mp@subsup{S}{}{\prime}.\mp@subsup{S}{}{\prime}\not={}\longrightarrow2*r<dist c s) using IH inv-def by
simp
            then show ?thesis by (smt case-assm s-def assms(1) dist-self dist-triangle3
singletonD)
    next
                assume }\mp@subsup{c}{1}{}=s\wedge\mp@subsup{c}{2}{}=
            then have False using local-assms(3) by simp
            then show?thesis by simp
        qed
    qed
    moreover
```

    have \(\left(\forall c \in C \cup\{s\} . \forall s^{\prime \prime} \in S^{\prime}-\left\{s^{\prime} \in S^{\prime}\right.\right.\). dist \(\left.s s^{\prime} \leq 2 * r\right\}\).
            \(S^{\prime}-\left\{s^{\prime} \in S^{\prime}\right.\). dist \(\left.s s^{\prime} \leq 2 * r\right\} \neq\{ \} \longrightarrow 2 * 2<\) dist \(\left.c s^{\prime \prime}\right)\)
    using \(I H\) inv-def by fastforce
    moreover
have $\left(S^{\prime}-\left\{s^{\prime} \in S^{\prime}\right.\right.$. dist $\left.\left.s s^{\prime} \leq 2 * r\right\}=S \vee C \cup\{s\} \neq\{ \}\right)$ by simp
ultimately show ?thesis unfolding inv-def by blast
qed
lemma inv-last-1:
assumes $\forall s \in\left(S-S^{\prime}\right)$. distance $C s \leq 2 * r$
and $S^{\prime}=\{ \}$
shows radius $C \leq 2 * r$
by (metis Diff-empty assms image-iff radius-contained)
lemma inv-last-2:
assumes finite $C$
and card $C>n$
and $C \subseteq S$
and $\forall c_{1} \in C . \forall c_{2} \in C . c_{1} \neq c_{2} \longrightarrow$ dist $c_{1} c_{2}>2 * r$
shows $\forall C^{\prime}$. card $C^{\prime} \leq n \wedge$ card $C^{\prime}>0 \longrightarrow$ radius $C^{\prime}>r$ (is ?P)
proof (rule ccontr)
assume $\neg$ ? P
then obtain $C^{\prime}$ where card- $C^{\prime}$ : card $C^{\prime} \leq n \wedge$ card $C^{\prime}>0$ and radius- $C^{\prime}$ :
radius $C^{\prime} \leq r$ by auto
have $\forall c \in C .\left(\exists c^{\prime} . c^{\prime} \in C^{\prime} \wedge\right.$ dist $\left.c c^{\prime} \leq r\right)$
proof
fix $c$
assume $c \in C$
then have $c \in S$ using $\operatorname{assms}(3)$ by blast
then have distance $C^{\prime} c \leq$ radius $C^{\prime}$ using finite-distances by (simp add:

```
radius-def)
    then have distance C' }c\leqr\mathrm{ using radius- C' by simp
    then show \exists}\mp@subsup{c}{}{\prime}.\mp@subsup{c}{}{\prime}\in\mp@subsup{C}{}{\prime}\wedge\mathrm{ dist c c'}\leqr using dist-lemmas
        by (metis card-C' card-gt-0-iff)
    qed
    then obtain f where f:\forallc\inC.fc\in\mp@subsup{C}{}{\prime}\wedge dist c (fc)\leqr by metis
    have \neginj-on f C
    proof
        assume inj-on f C
    then have card C'}\geq\mathrm{ card C using <inj-on f C> card-inj-on-le card-ge-0-finite
card-C'f by blast
    then show False using card- }\mp@subsup{C}{}{\prime}<n<card C> by linarith
    qed
    then obtain c1 c2 where defs: c1 \inC ^c\mathcal{L}\inC ^c1 \not= c2 ^fc1=fc2
using inj-on-def by blast
    then have *: dist c1 (f c1) \leqr^ dist c2 (f c1) \leqr using f by auto
    have 2 * r < dist c1 c2 using assms defs by simp
    also have ... \leq dist c1 (fc1) + dist (fc1) c2 by(rule dist-triangle)
    also have ... = dist c1 (f c1) + dist c2 (f c1) using dist-commute by simp
    also have \ldots.. \leq2*r using * by simp
    finally show False by simp
qed
lemma inv-last:
    assumes inv {} Cr
    shows (card C \leqk \longrightarrow radius C \leq2*r) ^(card C>k 
0^ card C'\leqk\longrightarrow radius C'>r))
    using assms inv-def inv-last-1 inv-last-2 finite-subset[OF - finite-sites] by auto
theorem Center-Selection-r:
    VARS (S' :: ('a :: metric-space) set) (C :: ('a :: metric-space) set) (r :: real) (s
:: 'a)
    {True}
    S':= S;
    C:= {};
    WHILE S'\not={} INV {inv S'Cr} DO
        s:=(SOME s. s \in S S';
        C:=C\cup{s};
        S':= S'-{s'. s'\in S'^dist s s}\mp@subsup{s}{}{\prime}\leq2*r
        OD
    {(card C \leqk\longrightarrow radius C\leq2*r)^(card C>k\longrightarrow (\forallC'. card C'>}>>>
card C'
proof (vcg, goal-cases)
    case (1 S' Cr)
    then show ?case using inv-init by simp
next
    case (2 S' Cr)
    then show ?case using inv-step by simp
```

```
next
    case (3 S'Cr)
    then show ?case using inv-last by blast
qed
```


### 6.2 The Main Algorithm

definition invar $::$ ('a $::$ metric-space) set $\Rightarrow$ bool where invar $C=(C \neq\{ \} \wedge$ card $C \leq k \wedge C \subseteq S \wedge$

$$
\left(\forall C^{\prime} .\left(\forall c_{1} \in C . \forall c_{2} \in C . c_{1} \neq c_{2} \longrightarrow \text { dist } c_{1} c_{2}>2 * \text { radius } C^{\prime}\right)\right.
$$

$$
\left.\left.\vee\left(\forall s \in S . \text { distance } C s \leq 2 * \text { radius } C^{\prime}\right)\right)\right)
$$

abbreviation some where some $A \equiv(S O M E$ s. $s \in A)$
lemma invar-init: invar $\{$ some $S\}$
proof -
let ?s $=$ some $S$
have $s$-in-S: ?s $\in S$ using some-in-eq non-empty-sites by blast
have $\{? s\} \neq\{ \}$ by simp
moreover
have $\{S O M E$ s. $s \in S\} \subseteq S$ using $s$-in- $S$ by simp
moreover
have card $\{S O M E$ s. $s \in S\} \leq k$ using non-zero- $k$ by simp
ultimately show ?thesis by (auto simp: invar-def)
qed
abbreviation furthest-from where
furthest-from $C \equiv(S O M E$ s. $s \in S \wedge$ distance $C s=\operatorname{Max}($ distance $C$ ' $S)$ )
lemma invar-step:
assumes invar $C$
and card $C<k$
shows invar $(C \cup\{$ furthest-from $C\})$
proof -
have furthest-from-C-props: furthest-from $C \in S \wedge$ distance $C$ (furthest-from $C$ )
$=$ radius $C$
using someI-ex[of $\lambda x . x \in S \wedge$ distance $C x=$ radius $C]$ radius-def2 radius-def
by auto
have $C$-props: finite $C \wedge C \neq\{ \}$
using finite-subset[OF - finite-sites] assms(1) unfolding invar-def by blast \{
have card $(C \cup\{$ furthest-from $C\}) \leq$ card $C+1$
using assms(1) C-props unfolding invar-def by (simp add: card-insert-if)

```
    then have card (C\cup{furthest-from C})<k+1 using assms(2) by simp
    then have card (C\cup{furthest-from C})\leqk by simp
}
moreover
```

have $C \cup\{$ furthest-from $C\} \neq\{ \}$ by simp
moreover
have $(C \cup\{$ furthest-from $C\}) \subseteq S$ using assms(1) furthest-from-C-props unfolding invar-def by simp

## moreover

have $\forall C^{\prime} .\left(\forall s \in S\right.$. distance $(C \cup\{$ furthest-from $C\}) s \leq 2 *$ radius $\left.C^{\prime}\right)$
$\vee\left(\forall c_{1} \in C \cup\{\right.$ furthest-from $C\} . \forall c_{2} \in C \cup\{$ furthest-from $C\} . c_{1} \neq c_{2}$
$\longrightarrow 2 *$ radius $C^{\prime}<$ dist $c_{1} c_{2}$ )
proof
fix $C^{\prime}$
have distance $C$ (furthest-from $C$ ) >2* radius $C^{\prime} \vee$ distance $C$ (furthest-from
$C) \leq 2 *$ radius $C^{\prime}$ by auto
then show $\left(\forall s \in S\right.$. distance $(C \cup\{$ furthest-from $C\}) s \leq 2 *$ radius $\left.C^{\prime}\right)$
$\vee\left(\forall c_{1} \in C \cup\{\right.$ furthest-from $C\} . \forall c_{2} \in C \cup\{$ furthest-from $C\} . c_{1} \neq$
$c_{2} \longrightarrow 2 *$ radius $C^{\prime}<$ dist $c_{1} c_{2}$ )
proof (elim disjE)
assume asm: distance $C$ (furthest-from $C$ ) $>2 *$ radius $C^{\prime}$
then have $\neg\left(\forall s \in S\right.$. distance $C s \leq 2 *$ radius $\left.C^{\prime}\right)$ using furthest-from-C-props
by force
then have $I H: \forall c_{1} \in C . \forall c_{2} \in C . c_{1} \neq c_{2} \longrightarrow 2 *$ radius $C^{\prime}<$ dist $c_{1} c_{2}$
using assms(1) unfolding invar-def by blast have $\left(\forall c_{1} \in C \cup\{\right.$ furthest-from $C\} .\left(\forall c_{2} \in C \cup\{\right.$ furthest-from $C\} . c_{1} \neq c_{2}$ $\longrightarrow 2 *$ radius $\left.C^{\prime}<\operatorname{dist} c_{1} c_{2}\right)$ )
using dist-ins[of C 2 * radius $C^{\prime}$ furthest-from C] IH C-props asm by simp then show? ?thesis by simp
next
assume main-assm: 2 * radius $C^{\prime} \geq$ distance $C$ (furthest-from $C$ ) have $\left(\forall s \in S\right.$. distance $(C \cup\{$ furthest-from $C\}) s \leq 2 *$ radius $\left.C^{\prime}\right)$ proof
fix $s$
assume local-assm: $s \in S$
then show distance $(C \cup\{$ furthest-from $C\}) s \leq 2 *$ radius $C^{\prime}$
proof -
have distance ( $C \cup\{$ furthest-from $C\}$ ) $s \leq$ distance $C s$
using distance-mono[of $C C \cup\{$ furthest-from $C\}] C$-props by auto
also have $\ldots \leq$ distance $C$ (furthest-from $C$ )
using Max.coboundedI local-assm finite-distances radius-def furthest-from-C-props by auto
also have $\ldots \leq 2 *$ radius $C^{\prime}$ using main-assm by simp
finally show ?thesis.

```
            qed
            qed
            then show ?thesis by blast
        qed
    qed
    ultimately show ?thesis unfolding invar-def by blast
qed
lemma invar-last:
assumes invar C and \negcard C<k
shows card C=k and card C'>0^ card C'\leqk\longrightarrowradius C\leq2 * radius C'
proof -
    show card C = k using assms(1, 2) unfolding invar-def by simp
next
    have C-props: finite C^C\not={} using finite-sites assms(1) unfolding invar-def
by (meson finite-subset)
    show card C'>0^card C'
    proof (rule impI)
        assume C'-assms: 0 < card (C' :: 'a set) ^ card C'
        let ?r = radius C'
    have (\forall\mp@subsup{c}{1}{}\inC.\forall\mp@subsup{c}{2}{}\inC.\mp@subsup{c}{1}{}\not=\mp@subsup{c}{2}{}\longrightarrow2*?r<dist c}\mp@subsup{c}{1}{}\mp@subsup{c}{2}{})\vee(\foralls\inS.distanc
Cs\leq2*?r)
            using assms(1) unfolding invar-def by simp
    then show radius C\leq2* ?r
    proof
```



```
            obtain s where s-def: radius C= distance Cs}\wedges\inS\mathrm{ using radius-def2
by metis
        show ?thesis
        proof (rule ccontr)
            assume contr-assm: ᄀ radius C\leq2 * ?r
            then have s-prop: distance Cs>2*?r using s-def by simp
            then have }\forall\mp@subsup{c}{1}{}\inC\cup{s}.\forall\mp@subsup{c}{2}{}\inC\cup{s}.\mp@subsup{c}{1}{}\not=\mp@subsup{c}{2}{}\longrightarrow\mathrm{ dist c}\mp@subsup{c}{1}{}\mp@subsup{c}{2}{}>2*2
?r>
            using C-props dist-ins[of C 2*?r s] case-assm by blast
        moreover
        {
            have s\not\inC
            proof
                    assume s}\in
                    then have distance Cs\leqdist s s using Min.coboundedI[of distance C
        ` S dist s s]
            by (simp add: distance-def C-props)
                also have ... = 0 by simp
                    finally have distance C s=0 using dist-lemmas(4) by (smt C-props)
                        then have radius-le-zero: 2 * ?r < 0 using contr-assm s-def by simp
                        obtain }x\mathrm{ where x-def: ?r = distance C' }\mp@subsup{C}{}{\prime}\mathrm{ using radius-def2 by metis
                obtain l where l-def: distance C' }\mp@subsup{C}{}{\prime}=\mathrm{ dist x l using dist-lemmas(3) by
```

```
(metis C'-assms card-gt-0-iff)
            then have dist xl=?r by (simp add: x-def)
            also have ... < 0 using C'-assms radius-le-zero by simp
            finally show False by simp
            qed
            then have card (C\cup{s})>k using assms(1,2) C-props unfolding
invar-def by simp
            }
            moreover
            have C}\cup{s}\subseteqS\mathrm{ using assms(1) s-def unfolding invar-def by simp
            moreover
            have finite ( }C\cup{s})\mathrm{ using calculation(3) finite-subset finite-sites by
auto
            ultimately have }\forallC.\mathrm{ card C s k^ card C>0 }\longrightarrow\mathrm{ radius C> ?r using
inv-last-2 by metis
            then have ?r > ?r using C'-assms by blast
            then show False by simp
        qed
        next
            assume }\foralls\inS\mathrm{ . distance Cs}\leq2*\mathrm{ radius C'
            then show ?thesis by (metis image-iff radius-contained)
        qed
    qed
qed
theorem Center-Selection:
VARS (C ::('a :: metric-space) set) (s :: ('a :: metric-space))
    {k\leq card S}
    C:= {some S};
    WHILE card C<kINV {invar C} DO
        C:=C\cup{furthest-from C}
    OD
    {card C = k^(\forall C'. card C'>0 ^ card C' 
C')}
proof (vcg, goal-cases)
    case (1 C s)
    show ?case using invar-init by simp
next
    case (2 C s)
    then show ?case using invar-step by blast
next
    case (3 C s)
    then show ?case using invar-last by blast
qed
end
end
```


## References

[1] R. Berghammer and M. Müller-Olm. Formal development and verification of approximation algorithms using auxiliary variables. In M. Bruynooghe, editor, Logic Based Program Synthesis and Transformation, LOPSTR 2003, volume 3018 of $L N C S$, pages 59-74. Springer, 2003.
[2] R. Berghammer and F. Reuter. A linear approximation algorithm for bin packing with absolute approximation factor $3 / 2$. Sci. Comput. Program., 48(1):67-80, 2003.
[3] R. Eßmann, T. Nipkow, and S. Robillard. Verified approximation algorithms. In N. Peltier and V. Sofronie-Stokkermans, editors, Automated Reasoning (IJCAR 2020), volume 12167 of $L N C S$, page 12167. Springer, 2020. https://doi.org/10.1007/978-3-030-51054-1_ 17.
[4] J. M. Kleinberg and É. Tardos. Algorithm Design. Addison-Wesley, 2006.

