ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάλυση του Bin Packing Problem

Κωνσταντίνος Γ. Δαβιλάς

Επιβλέπων: **Βασίλειος Ζησιμόπουλος,** Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2016

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Γ. Δαβιλάς

Όνομα Π. Επώνυμο
Α.Μ.: 1115200400241

Επιβλέπων: **Βασίλειος Ζησιμόπουλος,** Καθηγητής ΕΚΠΑ

ΠΕΡΙΛΗΨΗ

Στο πλαίσιο αυτής της εργασίας θα μελετήσουμε το NP-complete Bin Packing Problem. Δεδομένου ενός συνόλου αριθμών/βαρών και ενός συνόλου από κάδους (bins) δεδομένης χωρητικότητας, να βρεθεί ο ελάχιστος αριθμός bins που απαιτούνται, ώστε να περιέχονται όλα τα βάρη/αριθμοί σε bins και κανένα bin να μην ξεπερνά την χωρητικότητά του. Θα δούμε τις κυριότερες πρακτικές εφαρμογές του Bin Packing Problem. Θα αναφέρουμε και θα μελετήσουμε τους σημαντικότερους αλγόριθμους που χρησιμοποιούνται για την λύση του προβλήματος και τις σημαντικότερες εφαρμογές του. Επιπλέον θα μελετήσουμε το πρόβλημα του Bin Packing Problem σε αντιστοιχία με το MapReduce καθώς μπορεί να μας δώσει λύσεις στην μελέτη του προβλήματος σε κλίκες και θα δούμε ακόμα το πρόβλημα σε μονοπάτι όπου κάθε στοιχείο X_l οφείλει να βρίσκεται στον ίδιο κάδο με το X_{l+1} προτείνοντας κάποιους αλγόριθμους που λύνουν το πρόβλημα μελετώντας τα θετικά και τα αρνητικά τους.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Αλγόριθμοι, Αλγοριθμική Επιχειρησιακή ΄Ερευνα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ : Bin Packing, NP-δυσκολία, Ακριβείς Αλγόριθμοι, Προσεγγιστικοί Αλγόριθμοι, MapReduce

ABSTRACT

The purpose of this thesis is to study the NP-complete Bin Packing Problem. Given a set of numbers / weights and a set of bins (bins) of given capacity, the minimum number of bins required is to be found, in order the bins to contain all the weights / numbers and no bin exceeds its capacity. We will see the main practical applications of Bin Packing Problem. We will report and study the most important algorithms used for the solution of the problem and their most substantial applications. Furthermore we will study the problem of Bin Packing Problem in correspondence with the MapReduce as it can offer solutions to the Bin Packing Problem in cliques. Finally, we will examine the problem in a path where each element X_i must be in the same bucket with X_{i+1}, proposing some algorithms that could resolve the problem by analyzing their pluses and minuses.

SUBJECT AREA: Algorithms, Algorithmic Operations Research

KEYWORDS: Bin Packing, NP-Hardness, Exact Algorithms, Approximation Algorithms, MapReduce

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της πτυχιακής, ουσιαστικά έρχεται και η ολοκλήρωση του κύκλου σπουδών μου στο τμήμα της Πληροφορικής και Τηλεπικοινωνιών. Ως εκ τούτου θα ήθελα να εκφράσω την ευγνωμοσύνη μου σε όλους αυτούς που με βοήθησαν στην πορεία αυτή. Ξεκινώντας από τους καθηγητές και του συμφοιτητές που ήταν ανά πάσα στιγμή διαθεσίμου για να λύσουν τους προβληματισμούς που προέκυπταν μέσα από τον κύκλο σπουδών. Δεν θα μπορούσα πότε να παραλείψω την οικογένεια μου που με στήριξε απεριόριστα στο διάστημα αυτό.

Ιδιαίτερα θα ήθελα να ευχαριστήσω από του καθηγητές τον επιβλέποντα καθηγητή της παρούσας πτυχιακής τον κ. Ζησιμόπουλο, για την αμέριστη βοήθεια του στα πλαίσια της πτυχιακής και την άμεση ανταπόκριση του στους διάφορους προβληματισμούς που είχα κατά την διάρκεια της εκπόνησης της εργασίας

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή στον Bin Packing

Το πρόβλημα του bin packing αποτελεί ένα ενδιαφέρων ιστορικά πρόβλημα. Πρόβλημα στο οποίο αντικείμενα με διαφορετικές ποσότητες πρέπει να πακεταριστούν σε ένα πεπερασμένο αριθμό από bins (κάδους) σταθερής χωρητικότητας, έτσι ώστε κανένα bin να μην ξεπερνά την χωρητικότητα του και παράλληλα να χρησιμοποιηθούν τα ελάχιστα bins.

Το Bin Packing Problem μπορεί να συνδεθεί με μια άλλη πολύ σημαντική κατηγορία προβλημάτων επιχειρησιακής έρευνας. Το Bin Packing Problem μπορούμε να το δούμε σαν μια ιδιαίτερη περίπτωση του cutting stock problem. Όταν ο αριθμός των bins περιοριστεί στο 1 και κάθε αντικείμενο χαρακτηρίζεται από ποσότητα και αξία, το πρόβλημα της συμπλήρωσης ενός bin έτσι ώστε να περιέχει τα μεγαλύτερο δυνατό μέρισμα αξιών είναι γνωστό ως knapsack problem.

Υπάρχουν πολλές διαφοροποιήσεις του Bin Packing Problem όπως το 2D Bin Packing problem (Claudiaux 2007 [1] , Lodi 2002[2], Pisinger & Sigurd 2005[3]) , το 3D Bin Packing (Martello 2000 [4]), multiobjective packing (Liu 2008[5]), online packing (Babel 2004[6]),. dynamic bin packing (Coffman 1983[7], Epstein and Levy, 2010[8].), multi-stage bin packing(Puchinger and Raidl, 2007[9]),variable sized packing (Bang-Jensen & Larsen, 2012 [10]. ; Boyar and Favrholdt,2012[11] , Kang and Park, 2003 [12]), και άλλα διάφορα.

Από άποψη υπολογιστικής πολυπλοκότητας το πρόβλημα του Bin Packing είναι NP-hard. Παρά το γεγονός ότι είναι NP-hard μπορούμε να βρούμε λύσεις χρησιμοποιώντας διάφορους αλγόριθμους. Σε πολλές περιπτώσεις βέλτιστες λύσεις μπορούν να βρεθούν χρησιμοποιώντας αποτελεσματικούς ευρετικούς αλγορίθμους (Johnson 1974[13]). Για παράδειγμα ο First Fit αλγόριθμος, στον οποίο θα αναφερόμαστε ως FF μας δίνει μια γρήγορη λύση αλλά συχνά δεν είναι η βέλτιστη, βάζοντας τα αντικείμενα στο πρώτο bin στο οποίο χωράνε. Ο αλγόριθμος μπορεί να γίνει ποιο αποτελεσματικός ταξινομώντας εκ τον προτέρων τα αντικείμενα με φθίνουσα σειρά, γνωστό ως First Fit decreasingly, παρόλο που ούτε αυτό μας εγγυάται την βέλτιστη λύση και για μεγάλες λίστες κάνει τον αλγόριθμο πιο αργό, προστίθεται ουσιαστικά χρόνος για την ταξινόμηση. Είναι γνωστό παρόλα αυτά ότι υπάρχει σειρά αντικειμένων τέτοια ώστε ο FF να μας δώσει την βέλτιστη λύση.

Πολλοί εξελικτικοί εφευρετικοί αλγόριθμοι (Coffman 1983), περιλαμβάνοντας τους γενετικούς αλγόριθμους (Chan 2007[14]) και τους πρακτικούς αλγόριθμους σμήνους (swarm algorithms) επινοήθηκαν για να βρουν λύση στο Bin Packing Problem. Παρά το γεγονός ότι μερικά προβλήματα Bin Packing είναι NP-complete επιδέχονται προγραμματιστικής λύσης (Han 2010 [24]), ή προσεγγιστικής ευρετικής λύσης (Gendreau 2004 [15] ; Stawowy 2008 [16] .).

Εδώ γεννιέται το ερώτημα για να βρεθεί βέλτιστη λύση , όταν υπάρχουν προσεγγιστικές λύσεις υψηλής ποιότητας. Σε πολλές εφαρμογές είναι σημαντικό να βρεθεί η βέλτιστη λύση, συγκεκριμένα όταν ο αριθμός των bins πρέπει να είναι μικρός και κάθε επιπλέον bin να είναι ιδιαίτερος ακριβό (πχ μεταφορές , job scheduling CPU) η βέλτιστη λύση μειώνει δραστικά το κόστος. Επιπλέον όταν μπορούμε να διακρίνουμε ποια είναι η βέλτιστη λύση για ένα πρόβλημα , μπορούμε να υπολογίσουμε με μεγαλύτερη ακρίβεια την ποιότητα τον υπόλοιπων λύσεων. Ενώ δεν πρέπει να παραβλέπουμε ότι το βέλτιστο Bin Packing παραμένει μια ενδιαφέρουσα υπολογιστική πρόκληση που μπορεί να οδηγήσει σε γνώση χρήσιμη σε άλλα προβλήματα

1.2 Εφαρμογές του Bin Packing

Σε αυτή την ενότητα θα ασχοληθούμε με τις πρακτικές εφαρμογές του Bin Packing Problem στις διαφορετικές εκδοχές του, που το καθιστούν επίκαιρο και διαρκώς ενδιαφέρον πρόβλημα. Η πλέων εμφανείς εφαρμογές του Bin Packing έρχονται από τον χώρο της βιομηχανίας, όπου βρίσκει εφαρμογές στα διάφορα στάδια από την παραγωγή έως την διανομή.

Από την πρώτη κιόλας γραμμή συναρμολόγησης το ζητούμενο ήταν να βελτιώσουν την γραμμή καλύπτοντας διάφορους περιορισμούς. Ο επιθυμητός στόχος ήταν οι διεργασίες να συμβαίνουν σε διάφορους σταθμούς εργασίας με τέτοιο τρόπο ώστε τα παράγωγα του ενός, τα οποία ταυτόχρονα αποτελούσαν προϊόντα του άλλου, να μην ξεπερνά τον ρυθμό παράγωγης τους το ρυθμό κατανάλωσης τους (γνωστό ως Assembly Line Balancing Problem). Μας δίνονται διεργασίες διαφορετικών μεγεθών , που υπόκεινται σε δεσμεύσεις προτεραιότητας και χρονικών δεσμεύσεων και μπορεί να εκφραστεί ως BPP με περιορισμούς.

'Ενα ακόμα χαρακτηριστικό παράδειγμα είναι αυτό της κοπής αποθεμάτων από διάφορα υλικά όπως τα μέταλλα, που παρασκευάζονται σε σταθερά μεγέθη και διαμορφώνονται ανάλογα με την παραγγελία. Η παραγγελίες έρχονται σε τυχαία μεγέθη, οπότε το πρόβλημα που δημιουργείται είναι πως θα χρησιμοποιήσουμε τον ελάχιστο αριθμό από τα προκαθορισμένα τμήματα για να καλύψουμε την παραγγελία.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η εφαρμογή του Bin Packing Problem στον τομέα της αποθήκευσης και της διανομής , όπως στην φόρτωση των φορτηγών και των κοντέινερ. Τα αντικείμενα συχνά ποικίλουν σε μεγέθη και σε σχήματα ενώ πολλά πιθανών να απαιτούν συγκεκριμένο τρόπο τοποθέτησης (οριζόντια/κάθετα) ή ακόμα και συγκεκριμένες συνθήκες όπως η θερμοκρασία και η υγρασία, που καθιστούν την αποθήκευση και την μεταφορά τους ιδιαιτέρως κοστοβόρα, οδηγώντας σε επιθυμία για όσο το δυνατών μεγαλύτερη πληρότητα του χώρου.

Ο προγραμματισμός των project μιας εταιρίας θυμίζει το Bin Packing Problem σε πολλά σημεία. Το πρόβλημα της κατανομής τον εργασιών σε χρονοδιάγραμμα μπορεί

να εκφραστεί σαν Bin Packing Problem, οι διάφοροι αλγόριθμοι μπορούν να χρησιμοποιηθούν άκοπα σε αυτί την περίπτωση.

Η κατανομή του προϋπολογισμού έχει κοινά χαρακτηριστικά από το knapsack problem, όπου ένα συγκεκριμένο ποσό πρέπει να μοιραστεί σε διαφορετικές διεργασίες με διαφορετικό κόστος. Έχοντας κατά νου την ομοιότητα, μπορούμε να ωφεληθούμε από τις διαδικασίες του knapsack problem στην μακροπρόθεσμη και βραχυπρόθεσμη κατανομή του budget.

Αξιομνημόνευτες και ιδιαίτερου ενδιαφέροντος είναι οι εφαρμογές του Bin Packing Problem στο χώρο της πληροφορικής. Με τον όγκο τον δεδομένων που χρησιμοποιούμε τόσο στα δικά μας μέσα όσο και σε εφαρμογές cloud η όσο το δυνατόν καλύτερη κατανομή εγκυρότατα στο Bin Packing Problem. Ιδιαίτερα στο cloud storage η υπηρεσία παρέχεται από πληθώρα αποθηκευτικών οντοτήτων, υπευθύνων για την αποθήκευση κ την φύλαξη τον πληροφοριών. Οι πληροφορίες αποθηκεύονται σε σκληρούς δίσκους (συνήθως ίδιου μεγέθους, διότι συμφέρει στο κόστος και στην συντήρηση), αν δούμε τον δίσκο ως bin και τον όγκο των δεδομένων που πρέπει να αποθηκεύσουμε ως αντικείμενο το οποίο πρέπει να αποθηκευτεί εξ ολοκλήρου σε ένα δίσκο βρισκόμαστε ακριβός πάνω στις βασικές αρχές του Bin Packing με το κέρδος σε κόστος από την όσο το δυνατόν καλύτερη κατανομή να είναι κάτι παραπάνω από εμφανές. Ενώ άξια αναφοράς είναι η εφαρμογή του BPP σε άλλους τομείς όπως οι παρακάτω , resource allocation, packet routing, paged computer memory systems, storage, multiprocessor scheduling .

2. ΑΝΑΛΥΣΗ BIN PACKING PROBLEM

2.1 Μαθηματικός Σχηματισμός Bin Packing Problem

Σε αυτή την ενότητα θα δώσουμε την μαθηματική έκφραση του βασικού Bin Packing Problem. Το Bin Packing Problem μπορεί να οριστεί ως εξής. Έχουμε αρκετούς κάδους ή bins (του ίδιου μεγέθους), και επιθυμούμε να πακετάρουμε n αντικείμενα σε όσο το δυνατών λιγότερους κάδους. Το Bin Packing Problem είναι ένα NP-hard problem (Garey and Johnson,1979).

Η μαθηματική έκφραση του προβλήματος είναι η ακόλουθη:

Δεδομένου ενός bin S μεγέθους V και μιας λίστας από n αντικείμενα με μεγέθη $a_1, \ldots, a_n$ να πακετάρουμε, να βρεθεί ο άκαιρος αριθμός από bins ***B*** and a B-partition $S_1 \cup \cdots \cup S_B$ του συνόλου $\{1, \ldots, n\}$ τέτοιο ώστε $\sum_{i \in S_k} a_i \leq V$ για όλα τα $k = 1, \ldots, B.$ Η λύση είναι βέλτιστη αν έχει το ελάχιστο *B*. Η τιμή-B της βέλτιστης λύσης συμβολίζεται **OPT** παρακάτω. Ένας δυνατός συμβολισμός του προβλήματος είναι ο εξής :

minimize $B = \sum_{i=1}^{n} y_i$

subject to $B \geq 1,$

(a) $\sum_{j=1}^{n} a_j x_{ij} \leq V y_i, \qquad \forall i \in \{1, \ldots, n\}$ (b)

(b) $\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j \in \{1, \ldots, n\}$

(c) $y_i \in \{0,1\}, \qquad \forall i \in \{1, \ldots, n\}$

(d) $x_{ij} \in \{0,1\}, \qquad \forall i \in \{1, \ldots, n\} \, \forall j \in \{1, \ldots, n\}$

Όπου $y_i = 1$ αν το bin i χρησιμοποιείται και $x_{ij} = 1$ αν το αντικείμενο j πρόκειται να μπει στο bin i.

(a) Ο περιορισμός τίθεται για να βεβαιώσει ότι κανένα bin δεν ξεπερνά την χωρητικότητα του. Η μεταβλητή x_{ij} *είναι* 1 αν το αντικείμενο *j είναι πακεταρισμένο στο bin*

i, 0 αν όχι. (b) Περιορισμός ώστε όλα τα αντικείμενα να έχουνε πακεταριστεί. Οι περιορισμοί (c) και (d) για να έχουμε δυαδικές μεταβλητές

Αυτή η έκφραση του Bin Packing Problem είναι γνωστή ως μονοδιάστατο βασικό bin packing problem.

Από την στιγμή που το πρόβλημα είναι NP-hard, να βρούμε την βέλτιστη λύση είναι μια πολύ χρονοβόρα διαδικασία και στις περισσότερες περιπτώσεις δεν είναι το προτιμητέο, ειδικά για μεγάλου μεγέθους περιπτώσεις. Γιαυτό πολλές περιπτώσεις προτιμήσαμε λύσεις που μπορούν να βρεθούν χρησιμοποιώντας αποτελεσματικούς προσεγγιστικούς αλγορίθμους. Πριν δούμε αυτούς του αλγόριθμους καλό είναι να δούμε εν συντομία τι σημαίνει NP-hard και γιατί το Bin Packing Problem είναι NP-hard

2.2 N P -hardness

Στην υπολογιστική θεωρία πολυπλοκότητας, η NP είναι μια από τις πιο θεμελιώδεις κλάσεις πολυπλοκότητας. Η συντομογραφία NP αναφέρεται σε "μη-ντετερμινιστικό πολυωνυμικό χρόνο".Διαισθητικά, το NP είναι το σύνολο όλων των προβλημάτων απόφασης, για τα οποία οι περιπτώσεις όπου η απάντηση είναι "ναι" υπάρχει επιβεβαιωμένη αποδείξει για το γεγονός ότι η απάντηση είναι όντως «ναι». Πιο συγκεκριμένα, οι αποδείξεις αυτές πρέπει να είναι επαληθεύσιμες σε πολυωνυμικό χρόνο από μια ντετερμινιστική μηχανή Turing. Σε ένα ισοδύναμο επίσημο ορισμό, NP είναι το σύνολο προβλημάτων απόφασης, όπου οι "ναι" -περιπτώσεις μπορούν να γίνουν αποδεκτές σε πολυωνυμικό χρόνο από μια μη-ντετερμινιστική μηχανή Turing. Η ισοδυναμία των δύο ορισμών προκύπτει από το γεγονός ότι ένας αλγόριθμος σε μια τέτοια μη-ντετερμινιστική μηχανή αποτελείται από δύο φάσεις, η πρώτη η οποία αποτελείται από μια εικασία για τη λύση, η οποία δημιουργείται με ένα μη-ντετερμινιστικό τρόπο, ενώ η δεύτερη αποτελείται από ένα ντετερμινιστικό αλγόριθμο που επαληθεύει ή απορρίπτει την εικασία ως έγκυρη λύση στο πρόβλημα.

NP-hard προβλήματα είναι εκείνα τα προβλήματα τα οποία είναι τουλάχιστον τόσο δύσκολα όσο τα NP προβλήματα, δηλαδή, όλα τα προβλήματα NP μπορούν να αναχθούν (σε πολυωνυμικό χρόνο) σε αυτά. Τα NP-hard προβλήματα δεν χρειάζεται να είναι στο NP, δηλαδή, ότι δεν χρειάζεται να έχουν λύσεις ελεγμένες σε πολυωνυμικό χρόνο.

Για να δείξουμε ότι το Bin Packing Problem είναι NP-hard τα το ανάγουμε στο NP πρόβλημα του 2-Partition.

Ορισμός (2-Partition) Δεδομένων στοιχείων με τα μεγέθη s1 , . . . , Sn , δεν μπορούν να κατατμηθούν σε δύο ομάδες ίσου μεγέθους ;

Ορισμός (Bin Packing με αναγωγή στην μονάδα) Λαμβάνοντας υπόψη τα στοιχεία με τα μεγέθη s1 , . . . , Sn ∈ (0 , 1] , να πακέταριστουν σε κάδους έτσι ώστε ο αριθμός των κάδων ναι είναι όσο το δυνατόν μικρότερος, με κάθε κάδος να έχει μέγεθος 1.

Το Bin Packing είναι NP-hard. Δεν υπάρχει κανένας αλγόριθμος k-προσέγγιστικός με k < 3/2 (εκτός αν P = NP).

Απόδειξη. Ανάγουμε το Partition στον Bin Packing . Λαμβάνοντας υπόψη ένα παράδειγμα α_1, α_2 . . .α_n , του Partition , θέτουμε A = $\sum_{i=1}^{n} a_n$ και εξετάζουμε την περιπτωση του Bin Packing με $w_i = 2\alpha_i / A$. Είναι εύκολο να δούμε ότι η απάντηση στο παράδειγμα κατανομής/Partition είναι NAI , αν και μόνο αν ο ελάχιστος αριθμός των κάδων για το Bin Packing είναι 2 . Αν υπήρχε ένας αλγόριθμος k-προσέγγιστικός για το Bin Packing με k < 3/2 , στη συνέχεια, όταν το ακριβές ελάχιστο είναι 2 κάδοι , ο αλγόριθμος αυτός θα το βρείσκει πάντα , από τους 3 κάδους και μετά θα ήταν μια προσέγγιση με συντελεστή ≥ 3/2 . Έτσι, από την ως άνω μείωση , ο αλγόριθμος αυτός θα έλυνε ακριβώς το πρόβλημα του Partition σε πολυωνυμικό χρόνο , πράγμα που θα σήμαινε P = NP.

2.3 Αλγόριθμοι

Σε αυτό το σημείο θα μελετήσουμε τους κυριότερους απλούς αλγόριθμους που χρησιμοποιούνται για την λύση του Bin Packing Problem. Χωρίζουμε τους αλγόριθμους σε τρεις κατηγορίες, τους online αλγόριθμους, semi-online αλγόριθμους και τους offline αλγόριθμους. Οι online αλογοριθμοι εφαρμοζονται με οταν έρθει το πρώτο αντικείμενο και κάθε αντικείμενο που τοποθετείται δεν μπορεί να επανατοποθετηθεί. Οι semi-online αλγόριθμοι αφαιρούν τον περιορισμό όπου τα αντικείμενα δεν μπορούν να επανατοποθετηθούν . Επιτρέπουν τα συσκευασμένα αντικείμενα να κινηθούν μετά τη συσκευασία. Ο αριθμός τον μετακινήσεων εξαρτάται από τον σχεδιασμό του αλγορίθμου. Οι offline αλγόριθμοι περιλαμβάνουν την υπόθεση ότι όλα τα αντικείμενα έχουν φτάσει πριν ξεκινήσει ο αλγόριθμος και έτσι μπορούν να διαταχτούν πριν ξεκινήσει ο κάθε αλγόριθμος. Το αποτέλεσμα είναι να πάρουμε αλγόριθμους με καλύτερη worst-case performance σε σχέση με τους online και τους semi-online (E. Coffman, M. Garey, and D. Johnson. [17])

2.3.1 Online algorithms

2.3.1.1 Αλγόριθμος Next-Fit

Ο απλούστερος αλγόριθμος για μια προσεγγιστική λύση στο BPP είναι ο Next-Fit αλγόριθμος ,το οποίο θα συμβολίζουμε ως NF. Στον NF ακολουθείται η εξής διαδικασία. Το πρώτο αντικείμενο προς τοποθέτηση ανατίθεται στον κάδο 1. Έχοντας τα αντικείμενα 2,...,n κάθε ένα από αυτά ανατίθεται στον τρέχοντα κάδο, αν χωράει τοποθετείται, αλλιώς ανατίθεται στον επόμενο ο οποίος γίνεται με την σειρά του ο τρέχον. Η πολυπλοκότητα του αλγόριθμου είναι προφανώς *O(n)*. Είναι εύκολο να αποδειχθεί ότι για κάθε στιγμιότυπο του BPP η υπολογιστική αξία *NF(I)* ικανοποιεί το όριο

$$NF(I) \leq 2\, z(I).$$

Η βέλτιστη λύση για κάθε στιγμιότυπο I συμβολίζεται ως *z(I).Συχνά* συμβολίζεται και ως OPT από τα αρχικά του optimal solution. Έστω ότι έχουμε την κακή περίπτωση όπου κάθε αντικείμενο που ελέγχουμε δεν χωράει στον ίδιο κάδο με το προηγούμενο . Έχουμε ως B1 τον πρώτο κάδο που θα χρησιμοποιήσουμε B2 τον δεύτερο και πάει λέγοντας, έχουμε να κατανείμουμε n αντικείμενα που θα κατανεμηθούν το κάθε ένα σε διαφορετικό κάδο.

Ας μελετήσουμε 2 περιπτώσεις για το n.

Case 1: n είναι άρτιος:

$c(B_1) + c(B_2) \geq 1$

$+c(B_3) + c(B_4) \geq 1$

...

$+c(B_{n-1}) + c(B_n) \geq 1$

$\sum_{i=1}^{n} w_i \geq n/2 \Rightarrow$

$n \leq 2 \sum_{i=1}^{n} w_i \leq 2 \lceil \sum_{i=1}^{n} w_i \rceil \leq 2OPT$

Case 2: n είναι περιττός:

$c(B_1) + c(B_2) \geq 1$

$+c(B_3) + c(B_4) \geq 1$

...

$+c(B_{n-2}) + c(B_{n-1}) \geq 1$

$\sum_{i=1}^{n} w_i \geq (n-1)/2 + c(B_n)$

$\lceil \sum_{i=1}^{n} w_i \rceil \geq (n-1 + 2c(B_n)) \geq n$

$n \leq 2 \sum_{i=1}^{n} w_i \leq 2 \lceil \sum_{i=1}^{n} w_i \rceil \leq 2OPT$

Επιπλέον υπάρχουν περιπτώσεις για τις οποίες η αναλογία ***r*** ***τ***ου *NF(I)/OPT(I) είναι* κοντά στο 2, ενώ στο worst-case το *r του NF , r(NF)=2.* Να σημειωθεί ότι worst-case performance ratio ενός αλγόριθμου A ορίζεται ως ο μικρότερος πραγματικός αριθμός r(A) τέτοιος ώστε

$$A(I)/z(I) \leq r(A) \text{ για όλες τις εκδοχές } I$$

όπου το A(I) συμβολίζει την λύση που μας δίνει ο αλγόριθμος A.

2.3.1.2 Αλγόριθμος Harmonic-Fit

Harmonic Fit (HF): Ο HF χωρίζει τα αντικείμενα σε σετ ανάλογα με το μέγεθος τους. Ο αριθμός των σετ μπορεί να είναι οποιοσδήποτε θετικός ακέραιος μεγαλύτερος από ένα. Στη συνέχεια, ο NF εφαρμόζεται σε κάθε σετ χωριστά όπως στο παράδειγμα 2.3.1.

2.3.1.3 Αλγόριθμος First-fit

Ένας άλλος αλγόριθμος είναι ο First-Fit (FF), έστω ότι έχουμε τα αντικείμενα με αύξουσα σειρά δεικτών,αναθέτουμε το κάθε αντικείμενο στον πρώτο κάδο στον οποίο χωράει (σύμφωνα με την σειρά που τους χρησιμοποιήσαμε).Αν το αντικείμενο δεν χωράει σε κανένα από τους ήδη υπάρχοντες κάδους (bins), τότε πάμε σε νέο κάδο όπως στο παράδειγμα 2.3.1.

Εάν υποθέτουμε ότι υπάρχουν *n* αντικείμενα, όπου το μέγεθος κάθε αντικειμένου, *si*,είναι *αυστηρά* μεταξύ 0 και 1. Θέλουμε να τοποθετήσουμε τα αντικείμενα στον ελάχιστο αριθμό κάδων μεγέθους 1. Κάθε κάθος μπορεί να διατηρεί οσαδήποτε αντικείμενα, αρκεί το άθροισμα των μεγεθών τους να μην ξεπερνά τη χωρητικότητα του κάδου. Ο First Fit Heuristic μεταχειρίζεται τα αντικείμενα στη σειρά και τοποθετεί κάθε αντικείμενο στον πρώτο κάδο που αυτό χωράει. Έστω $S= \sum_{i=1}^{n} s_i$

Η χειρότερη δυνατή περίπτωση για το Bin Packing είναι κάθε αντικείμενο να έχει μέγεθος μεγαλύτερο του ½ . Αυτό γιατί τότε κάθε κάδος θα χωράει το πολύ ένα αντικείμενο. Άρα έχουμε $s_i>1/2$ για κάθε i.

Τότε όμως $S = \sum_{i=1}^{n} s_i > n/2$. Η τελευταία σχέση μας λέει ότι αν έχουμε $\lceil 2S \rceil$ το πολύ κάδους αρκούν στη περίπτωση όπου κάθε κάδος χωράει ένα και μόνο ένα στοιχείο (που είναι και η χειρότερη περίπτωση για το bin packing).

Έχει αποδειχθεί από Johnson, Demers, Ullman, Garey and Graham ότι

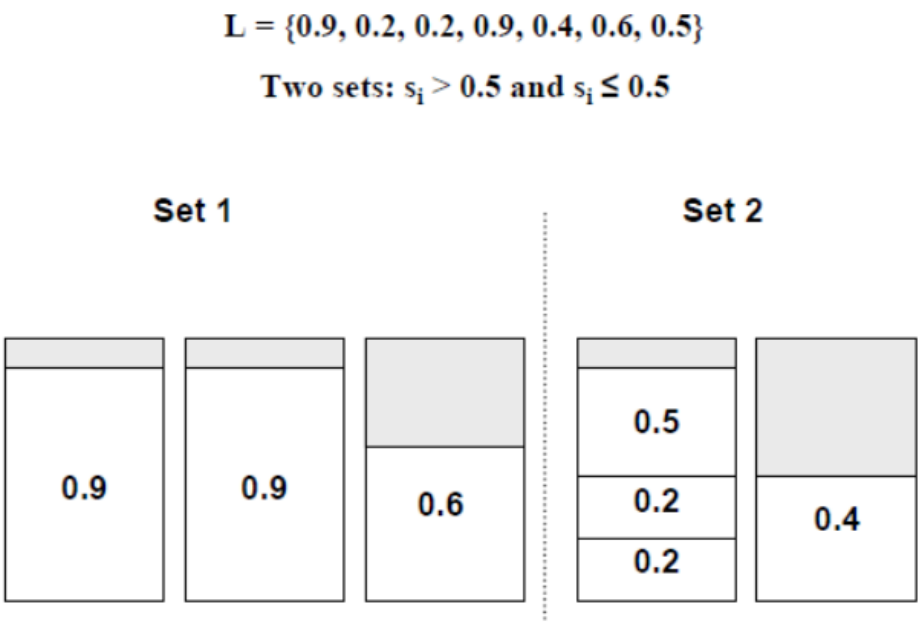
$$FF(I) \leq 17/10\ z(I) + 2 \quad (i)$$

για όλες τις εκδοχές I του BPP, και ότι υπάρχουν περιπτώσεις I με μεγάλο z(I) για τις οποίες $FF(I) \leq 17/10\ z(I) - 8$

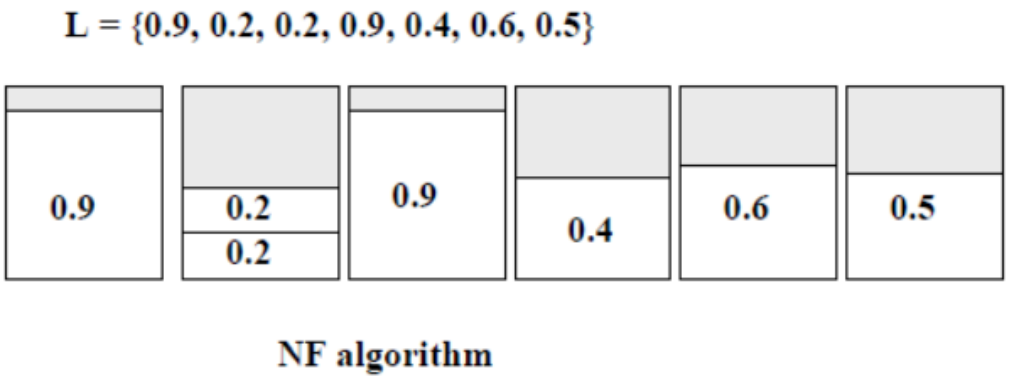
Εξαιτίας τις σταθεράς στο (I), καθώς και του αναλογικού αποτελέσματος τον άλλον αλγόριθμων, το worst-case performance ratio δεν μπορεί να μας δώσει πλήρεις πληροφορίες για την συμπεριφορά του worst-case. Ανταυτής , χρησιμοποιείται ευρέος η asymptotic worst-case performance ratio για το BPP. Για ένα προσεγγιστικό αλγόριθμο A, ορίζεται ως ο μικρότερος πραγματικός αριθμός $r^{\infty}(A)$ τέτοιο ώστε , για κάποιο θετικό ακέραιο k, $A(I)/z(I) \leq r^{\infty}(A)$ για όλες τις εκδοχές του I ικανοποιώντας το Z(I)>=k

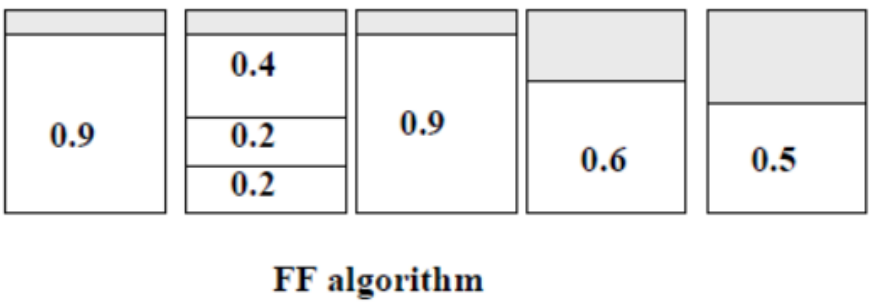
είναι προφανές από τα (i)-(ii), ότι r^{∞} (FF) = 17/10

Παράδειγμα 2.3.1 Έστω οτι έχουμε bins χωρητικότητας C=1 και πρέπει να κατανέμουμε , βάση των ανωτέρω αλγόριθμων τα στοιχεία του συνόλου L



Harmonic Fit algorithm for the BPP.





Next Fit and First Fit algorithms for the BPP.

2.3.1.4 Αλγόριθμος Best-Fit

Ο επόμενος αλγόριθμος είναι ο Best-Fit συμβολιζόμενος ως BF, ο αλγόριθμος BF προέρχεται από τον αλγόριθμο FF, τοποθετώντ

ας όμως το αντικείμενο προς τοποθέτηση στον διαθέσιμο κάδο (αν αυτός υπάρχει) με τον μικρότερο διαθέσιμο υπολειπόμενο χώρο. Οι Johnson, Demers, Ullman, Garey and Graham (1974) απέδειξαν οτι ισχύει το ίδιο worst-case όριο με τον FF δηλαδή

$$r^{\infty}(BF) = 17/10.$$

Η χρονική πολυπλοκότητα των NF και BF είναι *O* (*n* log*n)*. Αυτό επιτυνχάνεται χρησιμοποιώντας ένα 2-3 δεντρό όπου τα φύλλα του αποθηκεύουν την υπολειπόμενη χωρητικοτητα τον χρησιμοποιημένων κάδων.(Το δέντρο 2-3 είναι ένα δέντρο του οποίου : (a) κάθε κόμβος που δεν είναι φύλλο έχει 2 έως 3 παιδιά,(b) κάθε μονοπάτι από την ρίζα στον φύλλο έχει το ίδιο μήκος, (c) ταμπέλες στους κόμβους επιτρέπουν την αναζήτηση συγκεκριμένης αξίας φύλλου, αναβαθμίζοντας το ή εισάγοντας νέο φύλλο σε O(n) χρόνο. Περισσότερες πληροφορίες για την συγκεκριμένη δομή δεδομένων μπορούν να βρεθούν στο (Aho, Hopcroft and Ullman (1983)) Με αυτόν τον τρόπο κάθε προσέγγιση του FF ή του BF απαιτεί *O*(log*n*) χρόνο ,μιας και ο αριθμός των φύλλων περιορίζεται από το *n.*

Παράδειγμα 2.3.2

Έστω ότι έχουμε bins χωρητικότητας C=10 και πρέπει να κατανείμουμε βήμα-βήμα , βάση των ανωτέρω αλγόριθμων αντικείμενα του συνόλου L={ *3,6,2,1,5,7,2,4,1,9 }*

Βήμα	Next-Fit	First-Fit	Best-Fit
1	[3]	[3]	[3]
2	[3,6]	[3,6]	[3,6]
3	[3,6]-[2]	[3,6]-[2]	[3,6]-[2]
4	[3,6]-[2,1]	[3,6,1]-[2]	[3,6,1]-[2]
5	[3,6]-[2,1,5]	[3,6,1]-[2,5]	[3,6,1]-[2,5]
6	[3,6]-[2,1,5]-[7]	[3,6,1]-[2,5]-[7]	[3,6,1]-[2,5]-[7]
7	[3,6]-[2,1,5]-[7,2]	[3,6,1]-[2,5,2]-[7]	[3,6,1]-[2,5,2]-[7]
8	[3,6]-[2,1,5]-[7,2]-[4]	[3,6,1]-[2,5,2]-[7]-[4]	[3,6,1]-[2,5,2]-[7]-[4]
9	[3,6]-[2,1,5]-[7,2]-[4,1]	[3,6,1]-[2,5,2,1]-[7]-[4]	[3,6,1]-[2,5,2,1]-[7]-[4]
10	[3,6]-[2,1,5]-[7,2]-[4,1]-[9]	[3,6,1]-[2,5,2,1]-[7]-[4]-[9]	[3,6,1]-[2,5,2,1]-[7]-[4]-[9]

2.3.2 Offline Αλγόριθμοι

Ας υποθέσουμε ότι έχουμε λάβει όλα τα αντικείμενα και είναι ταξινομημένα με φθίνουσα σειρά έτσι ώστε $w_1 \geq w_2 \geq \ldots \geq w_n$

Αν εφαρμόσουμε σε αυτή την σειρά τους NF,FF,BF έχουμε ως αποτέλεσμα αλγόριθμους με χρονική πολυπλοκότητα O (nlogn), τους Next-Fit-Decreasing (NFD), First-Fit-Decreasing (FFD) και Best-Fit-Decreasing (BFD) . Η ανάλυση του worst-case για το NFD έγινε από τους Becker & Coffman (1981), ενώ για τα FFD και BFD από τους Johnson, Demers,Ullman,Garey & Coffman (1974), βασισμένη σε προηγούμενα αποτελέσματα του Johnson (1973) που απέδειξε ότι:

$$FFD(I) \leq 11/9\ z(I) + 4$$

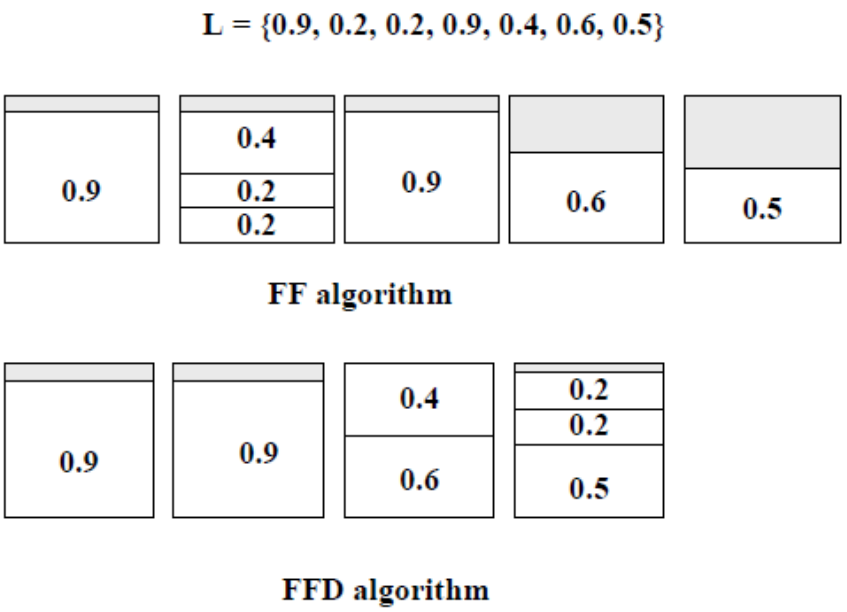
για κάθε εκδοχή του I. Τα αποτελέσματα συνοψίζονται στον παρακάτω πίνακα από το Coffman,Garey and Johnson (1984) στον οποίο οι τελευταίες τρεις στήλες δίνουν, για α=1/2, 1/3, 1/4 την τιμή r^{∞}_{α} του ασυμπτωτικού worst-case performance ratio των αλγόριθμων, όταν αυτοί εφαρμοστούν σε εκδοχές που ικανοποιούν ότι $\min_{1 \leq j \leq n} \{w_j\} \leq ac$

Asymptotic worst-case performance ratio of bin-packing algorithms

Αλγόριθμος	Χρονική πολυπλοκότητα	r^{∞}	$r^{\infty}_{1/2}$	$r^{\infty}_{1/3}$	$r^{\infty}_{1/4}$
NF	O(n)	2.000...	2.000...	1.500...	1.333...
FF	O(nlogn)	1.700...	1.500...	1.333...	1.250...
BF	O(nlogn)	1.700...	1.500...	1.333...	1.250...
NFD	O(nlogn)	1.691...	1.424...	1.302...	1.234...
FFD	O(nlogn)	1.222...	1.183...	1.183...	1.150...
BFD	O(nlogn)	1.222...	1.183...	1.183...	1.150...

Παράδειγμα 2.3.3

Σύγκριση FF με FFD για την κατανομή σε κάδους χωρητικότητας C=1 του τον αντικειμένων του συνόλου L



Παράδειγμα 2.3.4

Εφαρμόζοντας τους offline αλγορίθμους στο παράδειγμα το πρόβλημα διαμορφώνεται ως εξής έχουμε bins χωρητικότητας C=10 και πρέπει να κατανέμουμε, βάση των ανωτέρω αλγόριθμων το σύνολο L= { 3,6,2,1,5,7,2,4,1,9 }

Πρώτα κάνουμε το short το L σε {9,7,6,5,4,3,2,2,1,1}

Βήμα	Next-Fit D	First-Fit D	Best-Fit D
1	[9]	[9]	[9]
2	[9]-[7]	[9]-[7]	[9]-[7]
3	[9]-[7]-[6]	[9]-[7]-[6]	[9]-[7]-[6]
4	[9]-[7]-[6]-[5]	[9]-[7]-[6]-[5]	[9]-[7]-[6]-[5]
5	[9]-[7]-[6]-[5,4]	[9]-[7]-[6,4]-[5]	[9]-[7]-[6,4]-[5]
6	[9]-[7]-[6]-[5,4]-[3]	[9]-[7,3]-[6,4]-[5]	[9]-[7,3]-[6,4]-[5]
7	[9]-[7]-[6]-[5,4]-[3,2]	[9]-[7,3]-[6,4]-[5,2]	[9]-[7,3]-[6,4]-[5,2]
8	[9]-[7]-[6]-[5,4]-[3,2,2]	[9]-[7,3]-[6,4]-[5,2,2]	[9]-[7,3]-[6,4]-[5,2,2]

9	[9]-[7]-[6]-[5,4]-[3,2,2,1]	[9,1]-[7,3]-[6,4]-[5,2,2]	[9]-[7,3]-[6,4]-[5,2,2,1]
10	[9]-[7]-[6]-[5,4]-[3,2,2,1,1]	[9,1]-[7,3]-[6,4]-[5,2,2,1]	[9,1]-[7,3]-[6,4]-[5,2,2,1]

Όπως βλέπουμε παραπάνω, η ταξινόμηση είχε ως αποτέλεσμα ο FFD και ο BFD να χρειαστούν ένα bin λιγότερο από τους αντίστοιχους FF και BF.

2.4 Ακριβείς ή Exact Αλγόριθμοι

2.4.1 Εκτιμώμενος χαμένος χώρος ή L2 Lower Bound

Μια συνάρτηση υπολογισμού του κάτω ορίου του bin packing, που υπολογίζει αποτελεσματικά το κάτω όριο είναι απαραίτητη. Δωθούσης ενός κάτω ορίου για το bin packing problem, αν βρούμε μια λύση που χρησιμοποιεί τον ίδιο αριθμό από bins όπως το κάτω όριο γνωρίζουμε ότι έχουμε την βέλτιστη λύση και μπορούμε να σταματήσουμε τον έλενχο. Να προφανές κάτω όριο είναι το άθροισμα τον αντικειμένων διαιρούμενο με την χωρητικότητα των bins στρογγυλοποιημένα στον πάνω ακέραιο. Ένα καλύτερο όριο ξεκινά με το άθροισμα τον αντικειμένων και προσθέτει ένα εκτιμώμενο ποσό για την συνολική σπατάλη περιεκτικότητας στην κάθε λύση, πριν το διαιρέσει με την περιεκτικότητα. Αυτό είναι το L2 όριο των Martello and Toth (Martello & Toth 1990a,[18].1990b [19]) .

Έστω ότι **w** είναι ο αχρησιμοποίητος χώρος αρχικοποιημένος σε 0 , έχοντας περιεκτικότητα bins **C** και ένα σετ αντικειμένων ***S*** $(s_1,s_2,..,s_n)$. Όσο υπάρχουν αντικείμενα στο ***S***, αφαιρέσουμε το μεγαλύτερο αντικείμενο s_1 και υπολογίζουμε την υπολειπόμενη περιεκτικότητα $r = \boldsymbol{C} - s1$ που απομένει αφού το $s_{1\ τοποθετηθεί}$ στο bin. Τότε αφαιρούμε από το ***S*** *όλα* τα αντικείμενα με αξία ίση η μικρότερη από το *r* και αποθηκεύουμε το άθροισμα τους στην μεταβλητή *sum.* Αν το $sum \leq r$, προσθέτουμε το $r - sum$ στο **W.** Αν $sum > r$, τότε $sum - r$ is μεταφέρεται κ προστίθεται στο *sum* που υπολογίζεται για το επόμενο bin .Ο αλγόριθμος τερματίζει όταν δεν υπάρχουν άλλα αντικείμενα στο ***S***. Τότε υπολογίζεται ως

$$L_2(S) = \text{πάνω ακέραιος } \left(\lceil 1/C(w + \sum_{i=1}^{n} s_i) \rceil \right)$$

Οι Bin Packing αλγόριθμοι από των Eilon και Christofides1971 [20] και Martello και Toth [1990a] εξετάζουν τα στοιχεία εισόδου , ένα την φορά και τα τοποθετούν σε κάδους . Αντίθετα ,ο αλγόριθμος bin completion [Korf , 2002 [21] , 2003 [22]] ελέγχει τους κάδους , ένα κάθε φορά και να αποδίδει ένα πλήρες σύνολο στοιχείων σε αυτούς.

2.4.2 Αλγόριθμος Martello and Toth

Ένας από του καλός αλγόριθμους για να βρούμε την βέλτιστη λύση του bin-packing problem μας δίνεται από τους Martello και Toth (Martello& Toth 1990a ; 1990b). Ο branch-and-bound αλγόριθμος τους είναι πολύπλοκος και θα περιγράψουμε μόνο τα κύρια σημεία του. Ο αλγόριθμος παίρνει τα αντικειμενική με φθίνουσα σειρά και τοποθετεί τα αντικείμενα με σειρά μέσα σε κάθε ημησυμπληρωμένο bin όπου χωράνε , και σε ένα νέο bin, διακλαδώνοντας σε αυτές τις2 εναλλακτικές. Αυτό έχει ως αποτέλεσμα ο χώρος του προβλήματος να περιορίζεται στο n! , όπου το n είναι ο αριθμός τον αντικειμένων, άλλα αυτό είναι ένα <<κακό>> πάνω όριο, αφού πολλά αντικείμενα δεν θα χωρέσουν στο ίδιο bin όπως άλλα. Σε κάθε κόμβο του δέντρου αναζήτησης , Martello & Toth υπολογίζουν για τον first-fit, best-fit, και worst-fit decreasing την πληρότητα για την κάθε αντίστοιχη μερική λύση. Μια μερική λύση στο bin-packing problem είναι μια λύση όπου κάποια ,αλλά όχι όλα , τα αντικείμενα έχουν ανατεθεί σε bins. Η πληρότητα μιας μερικής λύσης, παίρνει τα τρέχοντα μερικώς-συμπληρωμένα bins, και αναθέτει τα υπόλοιπα, σε μη ανατεθειμένα αντικείμενα σε bins. Ο worst-fit decreasing αλγόριθμος τοποθετεί κάθε διαδοχικό αντικείμενο στον ημισυμπληρωμένο bin με την μεγαλύτερη απομείνουμε χωρητικότητα που μπορεί να χωρέσει την αξία του αντικειμένου. Κάθε μία από αυτές τις προσεγγιστικές λύση συγκρίνεται με το κάτω όριο στην εναπομείνασα λύση κ αποκαλείται *L3*. Το L3 όριο υπολογίζεται <<χαλαρώνοντας>> το υπολειπόμενο ψευδοπρόβλημα, αφαιρώντας το μικρότερο αντικείμενο και στην συνέχεια εφαρμόζοντας το L2 όριο σε κάθε μια από τις <<χαλαρωμένες>> περιπτώσει, επιστρέφοντας το μεγαλύτερο κάτω όριο. Το όριο L2 των Martello & Toth's περιορίζει ισάξια το εκτιμώμενο wasted-space όριο που περιγράφτηκε παραπάνω, άλλα χρησιμοποιεί ένα πιο περίπλοκο αλγόριθμο για να το υπολογίσει. Αν ο αριθμός των bins που χρησιμοποιήθηκαν από οποιοδήποτε από τις προσεγγιστικές περατώσεις ισούται με το κάτω όριο για την συμπλήρωση της αντίστοιχης μερικής λύσης, κανένας παραπάνω έλεγχος δεν γίνεται κάτω απο αυτό τον κόμβο . Αν ο αριθμός των bins σε κάθε προσεγγιστική λύση ισούται με το κάτω όριο του αρχικού προβλήματος , ο αλγόριθμος τερματίζει επιστρέφοντας την λύση ως βέλτιστη. Η κύρια πηγή αποτελεσματικότητας του αλγόριθμο Martello & Toth είναι ότι μειώνει το μέγεθος των εναπομεινάντων υποπροβλημάτων ,όπου θα συζητήσουμε παρακάτω στο κομμάτι “Σχέσεις Κυριαρχίας”.

2.4.3 Σχέσεις Κυριαρχίας

Μερικά σετ από αντικείμενα τοποθετημένα σε ένα bin οδηγούν εγγυημένα σε λύση τουλάχιστον το ίδιο καλή όσο αυτές που επιτυγχάνονταν τοποθετώντας άλλα σετ από αντικείμενα στο ίδιο bin. Θα ξεκινήσουμε με κάποια απλά παραδείγματα αυτών τον σχέσεων κυριαρχίας και στην συνέχεια θα πάμε στην γενικοποίηση τους. Αρχικά, έστω ότι έχουμε δύο στοιχεία x και y , τέτοια ώστε x+ y = c, όπου c είναι η χωρητικότητα του bin. Υποθέτουμε ότι στην βέλτιστη λύση, τα x y είναι σε διαφορετικά bins. Σε αυτή την περίπτωση μπορούμε να ανταλλάξουμε το y με όλα τα άλλα αντικείμενα που βρίσκονται στο ίδιο bin με το w x, χωρίς να αυξήσουμε τον αριθμό των bins. Αυτό μας δίνει μια νέα βέλτιστη λύση με τα x και y στο ίδιο bin. Επομένως, δεδομένου ενός

προβλήματος με δύο τιμές x και y τέτοιες ώστε x + y = c, μπορούμε πάντα να βάζουμε τα x και y στο ίδιο bin, με αποτέλεσμα ένα μικρότερο πρόβλημα (Gent 1998). Δυστυχώς αυτό δεν επεκτείνεται σε 3 η περισσότερα αντικείμενα που το άθροισμά τους είναι ακριβώς ίσο με την χωρητικότητα του bin.

Σαν ένα άλλο παράδειγμα, θεωρούμε ένα αντικείμενο x σαν το μικρότερο τέτοιο ώστε τα μικρότερα δύο (2) εναπομείναντα αντικείμενα προστιθέμενα στο x θα ξεπεράσουν το c. Δηλαδή το πολύ ένα επιπλέον αντικείμενο μπορεί να προστεθεί στο bin που βρίσκεται το x. Έστω ότι το y είναι το μεγαλύτερο εναπομείναν αντικείμενο τέτοιο ώστε x + y ≤ c. Τοτε ,βάζουμε το y στο ίδιο bin όπως το x χωρίς να χάνουμε ποιοτικά στην λύση μας. Ο λόγος είναι επειδή αν βάλουμε οποιοδήποτε άλλο αντικείμενο, έστω z, μαζί με το x, τότε θα μπορούσαμε να ανταλλάξουμε το y με το z, αφού z ≤ y και x + y ≤ c. Σαν τελευταίο παράδειγμα , ας υποθέσουμε ότι το y είναι το μεγαλύτερο από τα αντικείμενα που έχουν μείνει και μπορεί να προστεθεί στο x έτσι ώστε x+y ≤ c, υποθέτουμε ότι το y είναι ίσο η μεγαλύτερο του αθροίσματος οποιωνδήποτε σετ από τα εναπομείναντα αντικείμενα που μπορούν να προστεθούν στο x χώρις να ξεπερνούν το c. Σε αυτή την περίπτωση, μπορούμε πάλι να βάλουμε τα x και y στο ίδιο bin, χωρίς να χάνουμε από την ποιότητα της λύσης. Ο λόγος που συμβαίνει αυτό είναι ότι, οποιοδήποτε άλλο σετ από αντικείμενα που θα τοποθετούνταν στο ίδιο bin όπως το x θα μπορούσαμε να το ανταλλάξουμε με το y χωρίς να αυξήσουμε τον αριθμό των bins. Η γενική μορφή αυτής της σχέσης κυριαρχίας μας δίνεται από τους Martello & Toth (Martello & Toth 1990a; 1990b). Ορίζουμε ένα εφικτό σύνολο, σαν ένα σύνολο από αντικείμενα των οποίων το άθροισμα δεν μπορεί να ξεπεράσει την χωρητικότητα του bin. Με A και B να είναι δύο δυνατά σύνολα. Αν όλα τα αντικείμενα του B μπορούν να πακεταριστούν σε σετ bins των οποίων οι χωρητικότητες είναι αντικείμενα του A, τότε το σύνολο A κυριαρχεί του συνόλου B. Δεδομένων όλων τον δυνατών συνόλων που περιέχουν ένα κοινό αντικείμενο x, μόνο το κυριαρχών σύνολο χρειάζεται να να υπολογίζεται για ανάθεση στο bin που περιέχει x. Martello &Toth χρησιμοποιούν αυτή την σχέση κυριαρχίας για να μειώσουν το μέγεθος τον υποπροβλημάτων που προκύπτουν από την ερευνά τους . Δεδομένου ενός μερικώς λυμένου προβλήματος, όπου κάποια αντικείμενα έχουν ήδη ανατεθεί σε bins, οι Martello & Toth πρώτα μετατρέπουν το μερικώς λυμένο πρόβλημα σε ένα ισοδύναμο αρχικό πρόβλημα, όπου κανένα αντικείμενα δεν έχει ανατεθεί σε bin, σε δύο βήματα. Αρχικά, κάθε αντικείμενο που δεν έχει ανατεθεί , ή έχει ανατεθεί σε bin χωρίς άλλα αντικείμενα αφήνεται ως έχει . Εν συνεχεία, για κάθε bin που έχει παραπάνω από ένα αντικείμενο, όλα τα αντικείμενα που βρίσκονται σε αυτό αντικαθίστανται με ένα αντικείμενο ίσο με το άθροισμα τους, Εγκυόμενοι ότι κάθε ένα από τα αντικείμενα , που είχαν μπει στο ίδιο bin θα μείνουν μαζί. Για να μειωθεί το μέγεθος του προβλήματος, οι Martello & Toth παίρνουν κάθε αντικείμενο *x* με την σειρά,ξεκινώντας με το μεγαλύτερο αντικείμενο , και ελέγχοντας να δουν αν υπάρχει αν υπάρχει μοναδικό σύνολο από τρία (3) ή λιγότερα αντικείμενα , συμπεριλαμβανομένου του *x*, που να κυριαρχεί όλων των δυνατών συνόλων που περιλαμβάνουν το *x*. Εάν υπάρχει, βάζουν το *x* μαζί με αυτά τα αντικείμενα στο ίδιο bin, και αναδρομικά εφαρμόζουν τον ίδιο αλγόριθμο μείωσης για να φτάσουν σε μειωμένο πρόβλημα. Επίσης να χρησιμοποιούν τις σχέσεις κυριαρχίας για να περικόψουν κάποιες τοποθετήσεις των στοιχείων σε bins.

2.4.4 Αλγόριθμος Bin-Completion Korf

Ο καλύτερος μέχρι στιγμής αλγόριθμος για bin-packing, μας δίνεται από τον Korf (Korf 2002; Korf 2003) που τον ονόμασε *bin completion*, ο αλγόριθμος αυτός κάνει ποιο αποδοτική τη χρήση των σχέσεων κυριαρχίας . Όπως ο αλγόριθμος Martello και Toth ,

ο bin completion είναι επίσης ένας αλγόριθμος διακλάδωσης και οριοθέτησης, αλλά ερευνά ένα διαφορετικό κομμάτι του προβλήματος. Αντί να εξετάζει κάθε στοιχείο με τη σειρά του , και να αποφασίζει σε ποιο κάδο θα το τοποθετήσει , ερευνά τον κάθε κάδο με τη σειρά του , και εξετάζει τα μη-κυριαρχούμενα εφικτά στοιχεία που θα μπορούσαν να χρησιμοποιηθούν για την συμπλήρωση αυτού του bin. Ταξινομεί τα στοιχεία με φθίνουσα σειρά μεγέθους ,και να εξετάζει τους κάδους που περιέχουν κάθε στοιχείο με τη σειρά του , απαριθμώντας όλες τις μη-κυριαρχούμενες συμπληρώσεις αυτού του bin , και τις διακλαδώσεις , αν υπάρχουν περισσότερες από μία . Με άλλα λόγια , πρώτα συμπληρώνει τον κάδου που περιέχει το μεγαλύτερο στοιχείο , στη συνέχεια να ολοκληρώσει τον κάδο που περιέχει το δεύτερο μεγαλύτερο στοιχείο, κλπ.

Παράδειγμα 2.4.1.

Για να δούμε τον αλγόριθμο, έστω ότι έχουμε να κατανείμουμε αντικείμενα με το ακόλουθα μεγέθη

{100, 98, 96, 93, 91, 87, 81, 59, 58, 55, 50, 43, 22, 21, 20, 15, 14, 10, 8, 6, 5, 4, 3, 1, 0},

σε bins με capacity 100. Το 100 καταλαμβάνει ένα bin από μόνο του και το 0 δεν καταλαμβάνει χώρο, οπότε το πρόβλημα μας γίνεται ως εξής

{98,96, 93, 91, 87, 81, 59, 58, 55, 50, 43, 22, 21, 20, 15, 14,10, 8, 6, 5, 4, 3, 1}.

Το μόνο στοιχείο που μπορεί να πάει με το 98 είναι το 1 , έτσι μπορούμε να τα βάλει μαζί , αφήνοντας

{96, 93, 91, 87, 81,59, 58, 55, 50, 43, 22, 21, 20, 15, 14, 10, 8, 6, 5, 4, 3}.

Τοποθετούμε το 96 και το 4 μαζί , αφού το άθροισμα τους ακριβώς καταλαμβάνει την χωρητικότητα του bin , αφήνοντας

{93, 91, 87, 81, 59, 58, 55, 50, 43, 22, 21, 20, 15, 14, 10, 8, 6, 5, 3}.

Δεδομένου ότι το άθροισμα των δύο μικρότερων από υπόλοιπα στοιχεία , 5 και 3 , υπερβαίνει την υπολειπόμενη χωρητικότητα του κάδου που περιέχει 93 , δεν μπορούμε να τοποθετήσουμε περισσότερα από ένα στοιχείο στο δοχείο , και επιλέγουμε το μεγαλύτερο τέτοιο στοιχείο, το 6, για να πάει με 93, αφήνοντας

{91, 87, 81, 59, 58, 55, 50, 43, 22, 21, 20, 15, 14, 10, 8, 5, 3}.

Μπορούμε να συμπληρώσουμε το bin που περιείχε το 91 με το 8 , 5 , 3 , ή 5 + 3. Αφού το 8 κυριαρχεί τα άλλα επιμέρους στοιχεία , καθώς επίσης και το άθροισμα του σύνολου 5 + 3 , τοποθετούμε το 8 μετο 91 , αφήνοντας

{87, 81, 59, 58, 55, 50, 43, 22, 21, 20, 15, 14, 10, 5, 3}.

Το bin που περιείχε το *87* μπορεί να συμπληρωθεί με τα *10, 5, 3, 10+3* or *5+3*. Όλα αυτά κυριαρχούνται από το 1*0+3*, έτσι τοποθετούμε το *10+3* με το *87*, αφήνοντας

{81, 59, 58, 55, 50, 43, 22, 21, 20, 15, 14, 5}.

Μέχρι αυτό το σημείο, καμία διακλάδωσης δεν είναι απαραίτητη, αφού υπάρχει μόνο μία κυρίαρχη επιλογή για την συμπλήρωση κάθε ένα από τους κάδους που μελετήθηκαν μέχρι τώρα. Ο κάδος που περιέχει το 81 μπορεί να συμπληρωθεί με τα 15

, 14 , 5 , ή 14 + 5. Τόσο το 15 όσο και το 14 + 5 κυριαρχούν επί του 14 και του 5. Ωστόσο, ούτε το 15 , ούτε 14 + 5 κυριαρχούν ο ένας τον άλλον , έτσι και οι δύο πρέπει να θεωρηθούν προς τοποθέτηση , παράγοντας μια διακλάδωση δύο κατευθύνσεων στην αναζήτηση . Ευρηματικός , επιλέγουμε να εξερευνήσετε την συμπλήρωση με το μεγαλύτερο άθροισμα πρώτη , προσθέτοντας 14 + 5 στον κάδο με 81 , αφήνοντας

{59, 58, 55, 50, 43, 22, 21, 20, 15}.

Μπορούμε να συμπληρώσουμε το δοχείο που περιέχει το 59 με τα 22 , 21 , 20 , 15 , 22 + 15 , 21 + 20 , 21 + 15 , ή 20 + 15. Από αυτά, μόνο τα 22 + 15 και 21 + 20 είναι μη-κυριαρχούμενα από τα άλλα. Αυτό παράγει άλλη μια διακλάδωση δύο κατευθύνσεων στην αναζήτηση και επιλέγουμε την εναλλακτική λύση με το μεγαλύτερο ποσό , 21 + 20 , για να τοποθετήσουμε πρώτα με το 59 , αφήνοντας

{58, 55, 50, 43, 22, 15}.

Για να συμπληρώσουμε τον κάδο με το 58 , υπάρχει μόνο μία επιλογή undominated , 22 + 15 , έτσι βάζουμε τα τρία αυτά στοιχεία μαζί , αφήνοντας

*{*55, 50, 43*}*.

Τα μόνα στοιχεία που θα μπορούσαν ενδεχομένως να τοποθετηθούν στο ίδιο bin με ένα στοιχείο του μεγέθους 58 ή μεγαλύτερο είναι εκείνα του μεγέθους 42 ή μικρότερο. Σε αυτό το σημείο , όλα τα στοιχεία μεγέθους 42 ή μικρότερου έχουν ήδη τοποθετηθεί σε ένα bin με ένα στοιχείο μεγέθους 58 ή μεγαλύτερο. Ενώ θα μπορούσαμε να αλλάξουμε κάποια από τα μικρότερα στοιχεία στα εν λόγω bins , καμία τέτοια αναδιάταξη δεν θα μπορούσε ενδεχομένως να μειώσει τον αριθμό των εν λόγω κάδων , διότι κάθε στοιχείο μεγέθους 58 ή μεγαλύτερο απαιτεί τον δικό του κάδου . Έτσι , δεν υπάρχει καμία ανάγκη να κάνουν πίσω σε ένα από τα σημεία διακλάδωσης που συναντήσαμε , και μπορούμε να συνεχίσουμε την αναζήτηση σαν όλες οι προηγούμενες αποφάσεις ήταν αναγκαστικές.

Η γενική μορφή αυτού του κανόνα είναι η εξής: Αν $x > c/2$, όπου *c είναι* η χωρητικότητα του bin , και όλα τα αντικείμενα $\leq c - x$ έχουν τοποθετηθεί σε bins που περιέχουν αντικείμενα $\geq x$, τότε κάθε διακλάδωση που δημιουργήθηκε γεμίζοντας bins με αντικείμενα $\geq x$ μπορεί να αγνοηθεί το υπόλοιπο πρόβλημα με{ 55 , 50 , 43 } απαιτεί δύο ακόμη κάδους , δίδοντας μία βέλτιστη λύση έντεκα κάδων , ως εξής:

*{*100*}*, *{*98, 1*}*, *{*96, 4*}*, *{*93, 6*}*, *{*91, 8*}*, *{*87, 10, 3*}*, *{*81, 14, 5*}*, *{*59, 21, 20*}*, *{*58, 22, 15*}*, *{*55, 43*}*, *{*50*}*.

2.4.5 Γενικός Αλγόριθμος

Γενικά ο αλγόριθμος bin-completion δουλεύει ως εξής . Πρώτα υπολογίζουμε την λύση BFD. Στην συνέχεια υπολογίζουμε ένα κάτω όριο για το συνολικό πρόβλημα χρησιμοποιώντας το όριο του χαμένου χώρου (wasted-space) που αναφέραμε παραπάνω. Αν το κάτω όριο ισούται με τον αριθμό τον bins της λύσης BFD τότε αυτή ειναι η βέλτιστη λύση. Αλλιώς αρχικοποιούμε την καλύτερη λύση μέχρι στιγμής για τη λύση BFD και ξεκινάμε έναν έλεγχο διακλάδωσης-ορίου αυστηρά για καλύτερες λύσεις. Μόλις οι μερικές/τμηματικές λύσεις χρησιμοποιούν τόσα bins όσα η βέλτιστη ολική λύση που έχει βρεθεί μέχρι στιγμής, κόβουμε την διακλάδωση του ελέγχου. Μελετάμε τα

αντικείμενα με φθίνουσα σειρά και φτιάχνουμε όλες τις μη-κυριαρχούμενες συμπληρώσεις του bin που περιέχει το εν' λόγο αντικείμενο. Αν δεν υπάρχουν συμπληρώσεις ή υπάρχει μοναδική μη-κυριαρχούμενη συμπλήρωση, συμπληρώνουμε το bin με αυτήν , με αυτόν το τρόπο και πάμε στο bin που περιέχει το επόμενο αντικείμενο. Αν υπάρχουν παραπάνω από μια μη-κυριαρχούμενες συμπληρώσεις , τις διατάσσουμε σε φθίνουσα σειρά συνολικού μεγέθους, κ μελετάμε το μέγιστο που χωράει αφήνοντας τα άλλα ως πιθανά σημεία μελλοντικής διακλάδωσης. Όποτε βρίσκουμε μια πλήρης λύση καλύτερη , από την μέχρι στιγμής λύση, ανανεώνουμε την βέλτιστη μέχρι στιγμής λύση.

Για ένα χαμηλότερο όριο για μια μερική λύση, χρησιμοποιούμε το άθροισμα όλων των στοιχείων, καθώς και τον χώρο που απομένει στα bins που συμπληρώθηκαν μέχρι στιγμής, διαιρούμενο με την χωρητικότητα των bin και στρογγυλοποιειμένο στον επόμενο μεγαλύτερο ακέραιο αριθμό. Αντίστοιχα, μπορούμε να προσθέσουμε τον αριθμό των bins που συμπληρώθηκαν στο άθροισμα των υπολοίπων στοιχείων, διαιρεμένο με την χωρητικότητα του bin και στρογγυλοποιημένο προς τον πάνω ακέραιο. Αυτό το κάτω όριο είναι πιο αποτελεσματικό από ότι το εκτιμώμενο όριο χαμένου χώρου που περιγράφεται παραπάνω, επειδή συμπεριλαμβάνει την πραγματική σπατάλη χώρου στους συμπληρωμένους κάδους. Επιπλέον, υπολογίζεται σε σταθερό χρόνο, απλώς συσσωρεύοντας/αθροίζοντας τις ποσότητες σπαταλημένου χώρου στους συμπληρωμένους κάδους. Έτσι, μπορούμε να αντικαταστήσουμε ένα γραμμικού χρόνου κάτω όριο συνάρτησης, με μια πιο ακριβή συνάρτηση σταθερού χρόνου, χωρίς καμία απώλεια στην δυνατότητα κλαδέματος . Αν το κάτω όριο για μια μερική λύση ισούται ή υπερβαίνει τον αριθμό των κάδων στην καλύτερη λύση μέχρι στιγμής, κλαδεύουμε την εξέταση της εν λόγω μερική λύση.

Ο περισσότερος χρόνος στον αλγόριθμό bin-completion δαπανάται στον υπολογισμό των μη-κυριαρχούμενων συμπληρώσεων ενός κάδου . Προκειμένου να γίνει πιο αποδοτικό δουλεύουμε ως εξής. Δεδομένου ενός στοιχείου x , βρίσκουμε πρώτα το μεγαλύτερο στοιχείο y για τα οποία $x + y \leq c$.

Στη συνέχεια υπολογίζουμε όλα τα εφικτά ζεύγη w και z , τέτοια ώστε $x + w + z \leq c$, τα οποία είναι μη-κυριαρχούμενα από κάθε μοναδικό y, δηλαδή $w + z > y$, και τέτοια ώστε κανένα ζεύγος να μην κυριαρχεί σε οποιοδήποτε άλλο.

Ας υποθέσουμε , χωρίς βλάβη της γενικότητας ότι $d > f$. Σε αυτήν την περίπτωση , το g πρέπει να είναι μεγαλύτερο από το e, ή $d + e$ να κυριαρχήσει στο $f + g$. Έτσι, δεδομένων οποιωνδήποτε δύο ζευγών στοιχείων, όπου κανένα ΔΕΝ κυριαρχεί το άλλο , το ένα πρέπει να εντελώς <<φωλιασμένο>> μέσα στο άλλο σε ταξινομημένη σειρά. Μπορούμε να δημιουργήσουμε όλα αυτά τα μη-κυριαρχούμενα ζεύγη σε γραμμικό χρόνο, κρατώντας όλα τα υπόλοιπα στοιχεία ταξινομημένα , έχοντας δυο δείκτες (pointers) πάνω στην λίστα,ένα δείκτη σε ένα μεγαλύτερο στοιχείο και ένα δείκτη σε ένα μικρότερο. Εάν το άθροισμα των δύο στοιχείων υπερβαίνει τη χωρητικότητα του bin , θα μετατοπίσουμε τον μεγαλύτερο δείκτη προς τα κάτω στο αμέσως επόμενο μικρότερο στοιχείο . Εάν το άθροισμα των δύο (δικτούμενων) στοιχείων είναι μικρότερο ή ίσο με το y, μετατοπίσουμε το μικρότερο δείκτη στον αμέσως μεγαλύτερο στοιχείο. Εάν καμία από αυτές τις περιπτώσεις δεν συμβεί , έχουμε ένα μη-κυριαρχούμενο ζευγάρι, και θα μετατοπίσουμε το μεγαλύτερο δείκτη προς τα κάτω, και το μικρότερο δείκτη επάνω. Σταματάμε όταν οι δείκτες φτάσουν ο ένας τον άλλον .

Μετά τον υπολογισμό όλων των μη-κυριαρχούμενων ζευγών συμπλήρωσης , μπορούμε στη συνέχεια να υπολογίσει όλες τις τριάδες d, e , f και , έτσι ώστε

$$x + d + e + f \leq c \text{ και } d + e + f > y ,$$

μετά όλες τις τετράδες , κλπ. Για να υπολογίσουμε όλες τις τριάδες , επιλέγουμε κάθε εφικτό πρώτο στοιχείο , και στη συνέχεια, χρησιμοποιούμε όλα τα μη-κυριαρχούμενα ζεύγη για τα υπόλοιπα δύο στοιχεία . Για όλες τις τετράδες, επιλέγουμε όλα τα πιθανά ζεύγη , τότε χρησιμοποιήστε όλα τα μη-κυριαρχούμενα ζεύγη για τα υπόλοιπα δύο στοιχεία , κλπ. Αυτά τα μεγαλύτερα εφικτά σύνολα, θα σε γενικές γραμμές περιλαμβάνουν κυριαρχούμενα σύνολα.

Δεδομένων δύο εφικτών συνόλων, ο καθορισμός του, εάν το μεγαλύτερο άθροισμα κυριαρχεί του άλλου είναι ένα άλλο ένα πρόβλημα bin packing . Αυτό το πρόβλημα είναι συνήθως τόσο μικρό που το λύσουμε άμεσα με brute-force search.

2.5 Αναλογία Bin packing- MapReduce

Ένα ενδιαφέρον πρόβλημα, για μελέτη παράλληλα με το BPP είναι αυτό του MapReduce, καθώς βασικοί αλγόριθμοι του bin packing χρησιμοποιούνται για την εύρεση προσεγγιστικής λύσης του MapReduce. Ενώ αντίστροφα , οι αλγόριθμοι που χρησιμοποιούνται στο πρόβλημα του MapReduce, μας δίνουν ταυτόχρονα λύση σε συγκεκριμένες εκδοχές του Bin Packing. Παρακάτω θα δούμε τι είναι εν γένη το MapReduce, και σε ποιες περιπτώσεις του Bin Packing μπορεί να μας είναι χρήσιμο.

2.5.1 Εισαγωγή στο MapReduce

Το MapReduce είναι ένα σύστημα προγραμματισμού που χρησιμοποιείται για παράλληλη επεξεργασία των δεδομένων μεγάλης κλίμακας. Έχει δύο φάσεις , τη φάση χαρτογράφησης (Mapping Schema) και τη φάση μείωσης . Τα δεδομένα εισόδου επεξεργάζονται από την φάση χαρτογράφησης που ορίζετε από τον χρήστη για να παράγουν ενδιάμεσα δεδομένα (της μορφής key, value) .Τα ενδιάμεσα δεδομένα , στη συνέχεια , υποβάλλονται σε επεξεργασία από τη φάση μείωσης που εφαρμόζει μια συνάρτηση οριζόμενη από το χρήστη στα κλειδιά και στις σχετικές τιμές τους. Το τελικό αποτέλεσμα παρέχεται από την φάση μείωσης

Ένας αλγόριθμος MapReduce μπορεί να περιγραφεί από ένα σχήμα χαρτογράφησης , η οποία αποδίδει τις εισόδους σε ένα σύνολο από reducers , έτσι ώστε για κάθε απαιτούμενη έξοδο να υπάρχει ένας reducer που λαμβάνει υπόψιν όλες τις εισόδους που συμμετέχουν στον υπολογισμό της η εξόδου αυτής . Ωστόσο , μεμονωμένες είσοδοι μπορεί να διαφέρουν όσον αφορά το μέγεθος τους . Το κόστος επικοινωνίας μπορεί να βελτιστοποιηθεί με την ελαχιστοποίηση του αριθμού αντιγράφων των εισόδων που αποστέλλονται στους reducers . Το κόστος επικοινωνίας είναι στενά συνδεδεμένο με τον αριθμό των reducers συγκεκριμένης περιεκτικότητας που χρησιμοποιούνται για να φιλοξενήσει τις εισόδους.

Στο MapReduce συναντάμε δύο βασικές οικογένειες προβλημάτων. Μία οικογένεια προβλημάτων δυο διαφορετικών συνόλων X και Y όπου κάθε στοιχείο του συνόλου X πρέπει να τοποθετηθεί σε reducer με όλα τα στοιχεία του συνόλου Y , γνωστή ως X2Y, που σχηματικά μπορεί να αναπαρασταθεί με ένα γράφο. Και μία άλλη οικογένεια προβλημάτων όπου απαιτείται ότι η κάθε εισόδος συναντά τις υπόλοιπες εισόδους σε ένα τουλάχιστον reducer, η οποία είναι γνωστή ως all-2-all περίπτωση συμβολιζόμενη

ως A2A. Σχηματικά αυτό μπορούμε να το προσομοιάσουμε με μια κλίκα όπου κάθε σημείο της πρέπει να πακεταριστεί τον ίδιο reducer με κάθε ένα από τα στοιχεία που επικοινωνεί τουλάχιστον μία φορά, πράγμα ταυτόσημο με το bin packing σε κλίκα.

2.5.2 Απόδοση MapReduce

Ένα σημαντικό μέτρο απόδοσης για αλγορίθμους MapReduce είναι η ποσότητα των δεδομένων που μεταφέρονται από τους χαρτογράφους (οι διαδικασίες που εφαρμόζεται η λειτουργία χαρτογράφησης) στους reducers (η διαδικασίες που εφαρμόζουν τη λειτουργία μείωσης) . Αυτό ονομάζεται κόστος επικοινωνίας
Ορίζουμε χωρητικότητα reducer να είναι το ανώτατο όριο μεγέθους αξιών που μπορεί να εκχωρηθεί στον reducer. Στα πλαίσια αυτής της εργασίας θεωρούμε ότι όλοι οι reducer έχουν την ίδια χωρητικότητα.
Οι ακόλουθες σχέσεις εμφανίζονται σε αλγόριθμους MapReduce
Μια αντίστροφη σχέση μεταξύ της χωρητικότητας reducer και τον συνολικό αριθμό των reducer. Για παράδειγμα , μεγάλης χωρητικότητας reducer επιτρέπει τη χρήση ενός μικρότερου αριθμού reducers. Μια αντίστροφη σχέση μεταξύ της χωρητικότητας reducer και τον παραλληλισμό . Για παράδειγμα, αν θέλουμε να επιτύχουμε υψηλό βαθμό παραλληλισμού, θέλουμε χαμηλής χωρητικότητας reducers. Μια αντίστροφη σχέση μεταξύ της χωρητικότητας reducer και το κόστος επικοινωνίας . Για παράδειγμα, στην περίπτωση που η χωρητηκότητα του reducer είναι ίση με το συνολικό μέγεθος των δεδομένων, τότε μπορούμε να χρησιμοποιήσουμε ένα reducer και να έχουν το ελάχιστο κόστος επικοινωνίας (φυσικά, αυτό πηγαίνει σε βάρος του παραλληλισμού) .

Το Mapping Schema είναι ένα σχήμα χαρτογράφησης όπου συμβαίνει η εκχώρηση του συνόλου των εισόδων σε reducers , έτσι ώστε οι ακόλουθοι δύο περιορισμοί να ικανοποιούνται :
α) Σε κάθε reducer έχουν ανατεθεί είσοδοι των οποίων το άθροισμα των μεγεθών είναι μικρότερο ή ίσο με τη χωρητικότητα του reducer q .
β) Για κάθε έξοδο , πρέπει να εκχωρηθούν οι αντίστοιχες είσοδοι του σε τουλάχιστον ένα κοινό reducer .

Ένα σχήμα χαρτογράφησης είναι βέλτιστο όταν το κόστος επικοινωνίας είναι ελάχιστο. Ο αριθμός των reducers που χρησιμοποιούμε συχνά είναι ελάχιστος για ένα βέλτιστο σχήμα χαρτογράφησης , αλλά αυτό δεν είναι πάντα το ζητούμενο . Είναι επιθυμητό να ελαχιστοποιείται και το μέγεθος του reducer επίσης. Το μέγεθος των reducers όπως αναφέραμε και νωρίτερα είναι σταθερό στα πλαίσια αυτής της μελέτης, αν διαιρέσουμε τόσο τη χωρητικότητα του reducer q όσο και τα μεγέθη των εισόδων με το μέγεθος q , μπορούμε να δούμε τους reducers ως bins μεγέθους 1 (ένα) και τις εισόδους ως στοιχεία προς εισαγωγή σε αυτά τα bins.

2.5.3 Αλγόριθμοι για το Mapping Schema

Έχει αποδειχθεί το το Mapping Schema είναι NP -hard τόσο για την A2A όσο και για την X2Y περίπτωση [23].Δεδομένου ότι το πρόβλημα A2A Mapping Schema είναι NP -hard , ψάχνουμε για ειδικές περιπτώσεις και αναπτύσσουμε αλγόριθμους για την λύση τους. Υπάρχοντες αλγόριθμοι βασίζονται στον bin packing, την επιλογή πρώτου αριθμού p και την κατανομή των εισροών σε δύο ομάδες με βάση τα μεγέθη τους. Οι α προσεγγιστικοί αλγόριθμοι έχουν δύο περιπτώσεις, ανάλογα με τα μεγέθη των εισόδων, ως εξής :

1. Είσοδος μεγεθών ≤ q/2
2 . Μία είσοδος είναι του μεγέθους w_i , μεγαλύτερο από το q/2, αλλά μικρότερο από το q , και όλες οι άλλες εισροές έχουν μέγεθος μικρότερο ή ίσο με q- w_i. Στην περίπτωση αυτή, το μεγαλύτερο μέρος του κόστους επικοινωνίας προέρχεται από την προσπάθια αντιστοίχισης της μεγάλης εισόδου με κάθε άλλη είσοδο .
Φυσικά , αν οι δύο μεγαλύτερες είσοδοι είναι μεγαλύτερες από το δεδομένο reducer χωρητικότητας q , τότε δεν υπάρχει λύση για το πρόβλημα σχήμα χαρτογράφησης A2A επειδή αυτές οι δύο είσοδοι δεν μπορουν να ανατεθούν σε ένα μόνο κοινό .

Παράμετροι για ανάλυση. Αναλύουμε τους προσεγγιστικούς αλγόριθμους για το Mapping Schema ανάλογα με τις ακόλουθες παραμέτρους :
1. Αριθμός reducers . Αυτός είναι ο αριθμός των reducers που χρησιμοποιείται από το σχήμα χαρτογράφησης για να στείλουμε όλες τις εισόδους
2. Το κόστος της επικοινωνίας , c . Το κόστος επικοινωνίας ορίζεται ως το άθροισμα όλων των bits που απαιτούνται ,
σύμφωνα με το σχήμα χαρτογράφησης, να μεταφέρει από τη φάση χαρτογράφησης στη φάση μείωσης.

2.5.4 Αλγόριθμος βασισμένος στο Bin Packing

Χρησιμοποιούμε ένα γνωστό αλγόριθμο bin packing να τοποθετήσει τις εισόδους m σε κάδους μεγέθους q/k. Ας υποθέσουμε ότι χρειαζόμαστε x κάδους για να τοποθετήσετε εισόδους m . Τώρα, κάθε ένας από αυτους τους κάδους θεωρείται ως μια ενιαία είσοδος του μεγέθους q/k για το πρόβλημά μας εύρεσης ενός βέλτιστου σχήματος χαρτογράφησης. Φυσικά, η παραδοχή είναι ότι όλες οι είσοδοι είναι μεγέθους το πολύ q/k. Οι FFD και BFD είναι οι πιο αξιοσημείωτοι αλγόριθμοι bin packing. Για παράδειγμα, ας δούμε την περίπτωση k=2 . Στην περίπτωση αυτή, δεδομένου ότι η χωρητικότητα κάθε reducer είναι q , κάθε δύο κάδοι μπορουν να ανατεθούν σε ένα μόνο reducer. Ως εκ τούτου, ο αλγόριθμος προσέγγισης χρησιμοποιεί το πολύ x (x-1)/2 reducers.

Από τη στιγμή που απαιτούνται το πολύ x (x-1)/2 reducers για μια λύση στο πρόβλημα χαρτογράφησης A2A , ο αλγόριθμος απαιτεί το πολύ $(11/9\ OPT\ )^2/\ 2$ reducers. Να σημειωθεί ότι , σε αυτή την περίπτωση , το OPT δεν δείχνει το βέλτιστο αριθμό απο reducers που ικανοποιούν το πρόβλημα A2A Mapping Schema. Το OPT δείχνει το βέλτιστο αριθμό των κάδων μεγέθους q/2 που απαιτούνται για να τοποθετήθουν οι m είσοδοι.

2.5.5 Αλγόριθμοι για ισομεγέθεις εισόδους

Όπως εξηγήσαμε παραπάνω, να ψάχνουμε για εισόδους του ίδιου μεγέθους είναι λογικό, διότι οι είσοδοι μέσω του bin packing τοποθετούνται σε κάδους του μεγέθους q/k , για k>=2 , και μόλις γίνει αυτό , μπορούμε να μεταχειριστούμε τους ίδιους τους κάδους ως εισόδους που θα σταλθούν στους reducers. Οπότε μπορούμε να δούμε την χωρητικότητα του reducer ως ένα πραγματικό αριθμό

Αναδρομικοί αλγόριθμοι, που όχι μόνο εξασφαλίζουν το όριο r ≤ (m-1)m/q-1 αλλά το κάνουν με τρόπους που χωρίζουν τους reducers σε «ομάδες» τ , όπου κάθε ομάδα έχει ακριβώς μία εμφάνιση της κάθε εισόδου, μας δίνονται από τους στο Aftati Ulman [23] συμβολιζόμενοι AU

Έχουν αποδείξει ότι με αυτούς τους αλγόριθμους μπορούμε να έχουμε βέλτιστα mapping schemas για τις εξής περιπτώσεις
1. q = 2
2. q = 3
3. q να είναι πρώτος και m = q^2
4. q-1 να είναι πρώτος και m = $(q-1)^2$ +q

Ενώ οι αλγόριθμοι αυτοί χωρίς να δίνουν βέλτιστα mapping schemas δίνουν καλά προσεγγιστικά αποτελέσματα και για τις υπολοιπές περιπτώσεις
Σε αυτή την ενότητα, παρέχουμε τους δύο αλγορίθμους για σχεδόν ίσου μεγέθους (q / k, όπου k> 1) εισόδους, για να εισέλθει κάθε ζεύγος εισόδων σε reducers χωρητικότητας q. Με άλλα λόγια, μας δίνονται πολλές είσοδοι μικρού μεγέθους σε σύγκριση με το q, εμείς πακετάρουμε τις εισόδους σε κάδους σταθερού μεγέθους της και στη συνέχεια να εξετάζουμε τον κάθε κάδο ως ενιαία είσοδο .

Οι δύο αλγόριθμοι μπορούν να συνοψιστούν ως εξής: Αλγόριθμοι 2-βημάτων (Αλγόριθμος 1 και Αλγόριθμος 2) πρώτα χειρίζονται τις εισόδους ώστε να οδηγηθούν σε σταθερά μεγέθη και στην συνέχεια κάθε ένας από τους αλγόριθμους, αναλαμβάνει ανάλογα αν oi reducers μπορεί να χωρέσει άρτιο ή περιττό αριθμό εισόδων. Οι αλγόριθμοι 1 και 2 χρησιμοποιούνται όταν το q είναι ένας άρτιος ή περιττός αριθμός αντίστοιχα.

Στόχος μας είναι να χρησιμοποιήσουμου τους Αλγόριθμους 1, 2 και να ελαχιστοποιηθεί το κόστος της επικοινωνίας μεταξύ της φάσης χάρτογράφιση και της φάσης μείωσης για ένα δεδομένο αριθμό εισόδων και reducers χωρητικότητας q.

Αλγόριθμος 1: Αλγόριθμος 2-βημάτων: όταν η χωρητικότητα του reducer q είναι ένας περιττός αριθμός. Για χάρη της κατανόησης και της παρουσίασης, παρουσιάζουμε ως παράδεγμα, reducer με q = 3, δηλαδή, ένα reducer μπορεί να κρατήσει το πολύ τρεις εισόδους. Ακολουθώντας το παράδειγμα δίνεται για q=3 , παρουσιάζουμε τον αλγόριθμο μας που χειρίζεται κάθε περιττή τιμή του q. Ο αλγόριθμος αποτελείται από τα πέντε παρακάτω στάδια:
Βήμα 1ο. Χωρίζουμε τις m εισόδους σε δύο σύνολα A και B μεγέθους
y = $\lfloor q/2 \rfloor$ ($\lfloor 2m/(q+1) \rfloor$ +1) και το χ = m - y, αντίστοιχα.

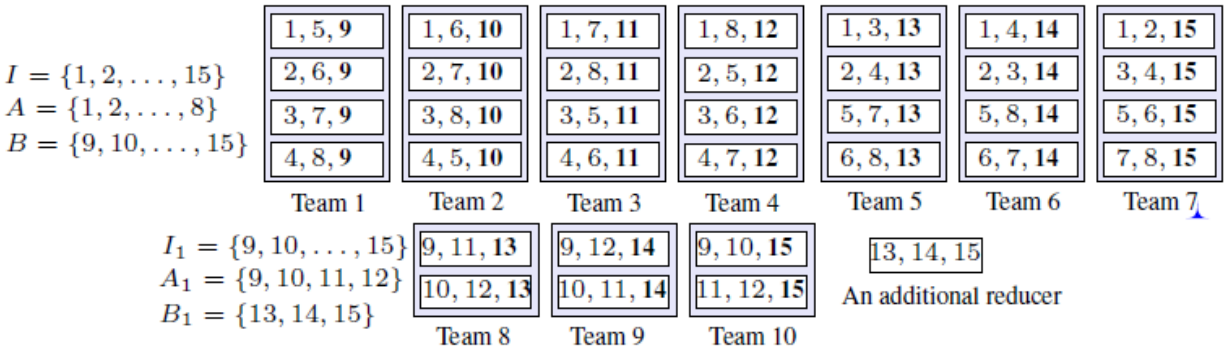
Βήμα 2ο. Γκρουπάρουμε τις εισόδους y σε u = $\lceil (y/q-\lceil q/2 \rceil) \rceil$ ξεχωριστές ομάδες, όπου κάθε ομάδα κρατά $\lceil (q-1)/2 \rceil$ εισόδους. (Θεωρούμε ότι κάθε μια από τις u= $\lceil (y/q-\lceil q/2 \rceil) \rceil$ ξεχωριστές ομάδες ως μία ενιαία είσοδο, όπου θα ονομάζουμε παρεχόμενη είσοδο. Φτιάχνοντας u ξεχωριστές ομάδες από τις y εισόδους του συνόλου A, θα μπορέσουμε να μετατρέψουμε την περίπτωση για κάθε περιττή τιμή του q σε μια περίπτωση όπου ένας reducer μπορεί να κρατήσει μόνο τρεις εισόδους, οι πρώτες δύο είσοδοι είναι ζεύγη των παρεχόμενων εισόδων και η τρίτη είσοδος είναι από το σύνολο B)

Βήμα 3ο. Οργανώνουμε τους (u - 1) X $\lceil u/2 \rceil$ reducers, ο καθένας χωρητικότητας q, με τη μορφή u - 1 ομάδων, με $\lceil u/2 \rceil$ reducers σε κάθε ομάδα. Αντιστοιχούμε κάθε δύο γρουπ σε ένα από τους (u - 1) X $\lceil u/2 \rceil$ reducers.

Βήμα 4ο. Όταν κάθε ζεύγος των παρεχόμενων εισόδων έχουν εκχωρηθεί, τότε αναθέτουμε τη i^{οστή} είσοδο του συνόλου B σε όλους τους reducers της i^{οστής} ομάδας.

Βήμα 5ο. Εφαρμόζουμε τα παραπάνω βήματα 1-4 για την ομάδα B, μέχρι να βρεθεί λύση στο πρόβλημα σχήμα χαρτογράφησης A2A για τις εισόδους x.

Παράδειγμα 2.5.1
'Εχουμε reducers με q = 3 και m = 15



Ο Αλγόριθμος 1, με q = 3, διαιρεί τις m εισόδους σε δύο ξένα μεταξύ τους σύνολα A και B y και x ≤ y-1 μεγέθους αντίστοιχα. Όταν q = 3, οργανώνουμε (y - 1) X $\lceil y/2 \rceil$ reducers (ο καθένας χωρητικότητας τρια) με τη μορφή y -1 ομάδων $\lceil y/2 \rceil$ reducers (ή παίκτων) σε κάθε ομάδα, και αυτοί οι reducers χρησιμοποιούνται για να ανατεθεί κάθε είσοδος του συνόλου A σε όλες τις εναπομείναντες εισόδους των συνόλων A και B. Σημειώστε ότι κάθε μια ομάδα πρέπει να έχει κάθε μία από τις εισόδους του συνόλου A ακριβώς μια φορά μεταξύ των reducers της ομάδας, το γεγονός αυτό θα είναι σαφές σύντομα.

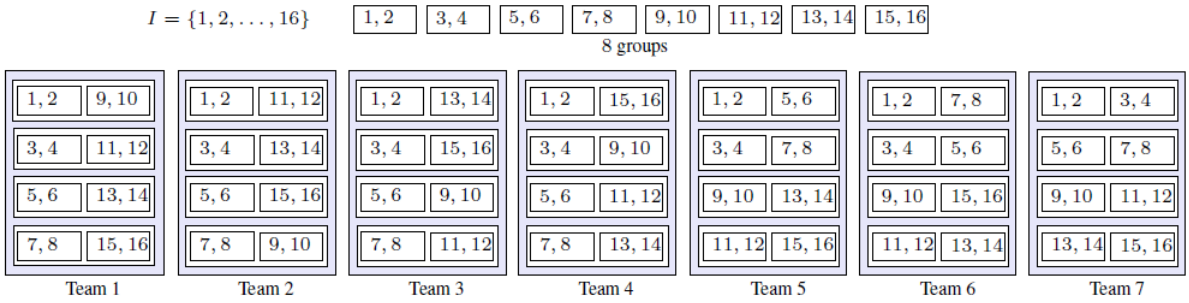
Υπάρχουν (y-1) X $\lceil y/2 \rceil$ ζεύγη των εισόδων του συνόλου A (κάθε ένα μεγέθους δύο) και υπάρχει ίδιος αριθμός reducers (κάθε ένας χωρητικότητας τριών) ως εκ τούτου, είναι δυνατόν να αναθέσουμε ένα ζεύγος σε κάθε reducer, και αυτές οι δύο είσοδοι γίνονται οι δύο από τις τρεις εισόδους που επιτρέπονται σε κάθε reducer. Μόλις, αναθέσουμε σε κάθε ζεύγος των εισόδων του συνόλου A στους (y-1) X $\lceil y/2 \rceil$ reducer, τότε θα αναθέτουμε της i^{οστή} είσόδο του συνόλου B σε όλους $\lceil y/2 \rceil$ reducer της i^{οστής} ομάδας. Περαιτέρω, ακολουθούμε μια παρόμοια διαδικασία για τις εισόδους του συνόλου B ώστε να ανατεθεί κάθε ζεύγος των υπολοίπων x εισόδων

Αλγόριθμος 2: Ο αλγόριθμος 2-βημάτων, για όταν η χωρητικότητα reducer q είναι ένας άρτιος αριθμός. Σας παρουσιάζουμε τον αλγόριθμο (βλέπε Αλγόριθμος 2) που χειρίζεται κάθε άρτια τιμή του q. Για χάρη της κατανόησης και παρουσίαση, παρουσιάζουμε πρώτα ένα παράδειγμα όπου q = 4, δηλαδή η περίπτωση κατά την οποία ένας reducer μπορεί να κρατήσει το πολύ τέσσερις εισόδους, όπως καταδεικνύεται στο σχήμα. Σημειώστε ότι σε αντίθεση με τον αλγόριθμο για περιττές τιμές του q (Αλγόριθμος 1) ο αλγόριθμος για άρτιες τιμές των q (Αλγόριθμος 2) δεν διαιρεί τις εισόδους m σε δύο ομάδες. Ο αλγόριθμος αποτελείται από δύο βήματα, ως εξής:

Βήμα 1ο. Γκρουπάρουμε τις m εισόδους σε u = $\lceil 2m/q \rceil$ ξεχωριστές ομάδες,

Βήμα 2ο. Οργανώνουμε τους (u-1) X (u / 2) reducer, ο καθένας χωρητικότητας q, με τη μορφή u-1 ομάδων u / 2 reducer σε κάθε ομάδα. Αναθέτουμε κάθε δύο ομάδες σε ένα από τα (u-1) X (u / 2) reducer.

Σημειώστε ότι θεωρούμε κάθε μία από τις u = $\lceil 2m/q \rceil$ ομάδες ως μία ενιαία είσοδο που ονομάζουμε παρεχόμενη είσοδο. Φτιάχνοντας u ξεχωριστές ομάδες από τις m εισόδους, μετατρέπουμε κάθε περίπτωση όπου το q είναι άρτιο ,σε μια περίπτωση, όπου το q = 2 (δηλαδή, ένας reducer μπορεί να κρατήσει μόνο δύο εισόδους) και να αναθέσουμε κάθε δύο παρεχόμενες εισόδους σε reducers. Με αυτόν τον τρόπο, κάθε είσοδος του συνόλου A αντιστοιχίζεται σε όλες τις υπόλοιπομενς m -1 εισόδους.
Παράδειγμα 2.5.2



'Εχουμε reducers με q = 4 και m = 16

2.6 BPP για κάθε διαδοχικό xi και xi+1 οφείλουν να πακετάρονται στο ίδιο bin

2.6.1 Ορισμός προβλήματος

Σε αυτό το σημείο θα μελετήσουμε μια διαφορετική εκδοχή του Bin Packing Problem προερχόμενη από τον προβληματισμό που έρχεται από το MapReduce στην περίπτωση που θέλουμε τα διαδοχικά στοιχεία x_i , x_{i+1} να συμπεριληφθούν στον ίδιο

reducer ,όπου οι reducers είναι δεδομένης χωρητικότητας C και θέλουμε την χρήση όσο τον δυνατόν λιγότερων reducers. Το πρόβλημα αυτό σε αναλογία με το BPP διαμορφώνεται ως εξής : δεδομένων κάδων με χωρητικότητα C και αντικειμένων x_i βάρους wi, να βρεθεί ο αριθμός των κάδων που απαιτούνται προκειμένου να τοποθετηθούν όλα τα αντικείμενα σε κάδους με τρόπο τέτοιο ώστε το άθροισμα των βαρών τον αντικειμένων σε κάθε κάδο να μην ξεπερνά το C και κάθε x_i , x_{i+1} να είναι τοποθετημένα σε ίδιο κάδο. Κάθε αντικείμενο μπορεί να τοποθετηθεί σε παραπάνω από ένα κάδο.

2.6.2 Κάτω Όριο

Όπως είναι εύκολα αντιληπτό το κάτω όριο $\sum_{i=1}^{n} xw_i$ /c συνεχίζει να αποτελεί κάτω όριο αλλά μπορεί να βελτιωθεί με τα την προσθήκη στο άθροισμα των επιπλέον στοιχείων. Γνωρίζουμε ότι θα χρειαστούμε τουλάχιστον $\lceil \sum_{i=1}^{n} xw_i / c \rceil$ κάδους για να κατανείμουμε τα αντικείμενα και κάθε κάδος θα δώσει ένα στοιχείο στον επόμενο, άρα κάθε κάθε κάδος πλην του τελευταίου θα συνεισφέρη κατά ένα επιπλέον βάρος, άρα αν $\lceil \sum_{i=1}^{n} xw_i / c \rceil$ = b, θα χρειαστεί να κατανεμηθούν σε κάδους n+(b-1) βάρη αντικειμενων. Στην ιδανικότερη περίπτωση αυτά τα b-1 επιπλέον βάρη είναι τα ελάχιστα . Αν πάρουμε το άθροισμα των w των b-1 ελαχίστων βαρών $\sum_{n=1}^{b-1} min\, xw_i$ ως το ελάχιστο άθροισμα επιπλέον βαρών από τα αντικείμενα που θα τοποθετηθούν σε 2 κάδους, θα οδηγηθούμε στο $\lceil (\sum_{i=1}^{n} xw_i + \sum_{n=1}^{b-1} min\, xw_i) / c \rceil$

2.6.3 Εφαρμογή Next-Fit στο προβλημα.

Θα ακολουθήσουμε το ίδιο σκεπτικό με τον Next-Fit για το κλασικό πρόβλημα, με την μόνη διαφορά να βρίσκεται όταν συμπληρώνεται ένα bin. Είναι προφανές ότι όταν η χωρητικότητα κάποιου bin y_i συμπληρωθεί με ένα αντικείμενο x_i με τέτοιο τρόπο ώστε να το x_{i+1} να μην χωράει στο y_i , για να έχουμε x_i και x_{i+1} , θα πρέπει να τοποθετήσουμε το x_i και στο bin y_i αλλά και στο bin y_{i+1} .

Παράδειγμα 2.6.1

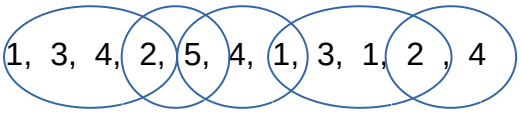
Για παράδειγμα έστω ότι έχουμε bin χωρητικότητας 10 και πρέπει να τοποθετήσουμε σε αυτά τα εξής αντικείμενα: *1,3,4,2,5,4,1,3,1,2,4*

Τοποθετώντας στον πρώτο κάδο θα φτάναμε στο σημείο 1,3,4,2 και θα είχαμε προς τοποθέτηση το *5,* επειδή το 5 δεν χώρα στο ίδιο κάδο , θα ξαναβάλουμε το 2 και στον 2ο κάδο παίρνοντας ως αποτέλεσμα :

1, 3, 4, 2, 5

bin 1 bin 2

Στην συνέχεια δεν χωράει το 4 ,οπότε επαναλαμβάνεται το 5 στον 3ο BIN και πάει λέγοντας, έχοντας ως αποτελέσματα τελικά



bin 1 bin 2 bin 3 bin 4 bin 5

Για να συμπληρώσουμε τους κάδους στην παραπάνω περίπτωση, ξεκινήσαμε από το πρώτο στοιχείο και βάζαμε διαδοχικά τα επόμενα στοιχεία, χρησιμοποιώντας ουσιαστικά ένα αλγόριθμο Next-Fit.

Συνοψίζοντας σε βήματα:

ΒΗΜΑ 1ο

Ξεκινώντας από τον πρώτο κάδο κ το πρώτο αντικείμενο x_1 , βάζουμε το αντικείμενο στον κάδο και συνεχίζουμε προσπαθώντας να βάλουμε το επόμενο x_{1+1} έως ότου φτάσουμε στο σημείο να μπει στον κάδο το τελευταίο αντικείμενο x_n κ να μην χωράει το αντικείμενο x_{n+1}

ΒΗΜΑ 2ο

Περνούμε νέο κάδο με πρώτο νέο στοιχείο, το τελευταίο στοιχείο (έστω x_n) του προηγούμενου κάδου κ συνεχίζουμε εισάγοντας το x_{n+1} και πάει λέγοντας έως ότου δεν χωράει άλλο στοιχείο στον κάδο

ΒΗΜΑ 3ο

Επαναλαμβάνουμε το βήμα 2ο έως ότου να τοποθετηθούν όλα τα αντικείμενα σε κάδους. Ο τερματισμός έρχεται με το τελευταίο αντικείμενο x_T . Ο αριθμός των κάδων που χρησιμοποιήθηκαν είναι η απάντηση του αλγορίθμου στο πρόβλημα

Σε σχέση με τον Next-Fit στο από Bin Packing Problem εδώ βλέπουμε να επαναλαμβάνεται ουσιαστικά ένα στοιχείο από κάθε κάδο στον επόμενο και συγκεκριμένα το τελευταίο στοιχείο κάθε κάδου.

Worst Case

Η χειρότερη περίπτωση που μπορούμε να έχουμε είναι να χωράνε δύο μόνο στοιχεία στον κάθε κάδο. Κάνοντας αναγωγή της χωρητικότητας στην μονάδα θα είχαμε ότι για κάθε τρία διαδοχικά αντικείμενα X_{k-1} , X_k , X_{k+1} τα βάρη τους w_{k-1} ,w_k , w_{k+1} >1

Άρα για n αντικείμενα θα είχαμε

$w_1 + w_2 + w_3 > 1$

$w_2 + w_3 + w_4 > 1$

...

$w_{n-2} + w_{n-1} + w_n > 1$

$$\sum_{i=1}^{n} w_i + \sum_{i=2}^{n-1} w_i + \sum_{i=3}^{n-2} w_i > n-2$$

$$3 \sum_{i=1}^{n} w_i + 2 > n$$

2.6.4 Αλγόριθμος maxRightLeft

Πριν ξεκινήσουμε την παρουσίαση του αλγορίθμου, καλό είναι να δούμε το κίνητρο που οδήγησε στην σύλληψη του, καθώς και τα χαρακτηριστηκά των αντικειμένων προς εισαγωγή. Βάζοντας αντικείμενα σε κάδους, ώστε τα διαδοχικά αντικείμενα να βρίσκονται στον ίδιο κάδο, περνούμε ως αποτέλεσα το τελευταίο στοιχείο ενός κάδου, να είναι ταυτόχρονα το πρώτο του επόμενου. Στην προσπάθεια μας να έχουμε όσο το δυνατόν μικρότερο αυτό το στοιχείο, θα ξεκινάμε από το μεγαλύτερο δυνατό αντικείμενο, από άποψη βάρους, κινούμενη αριστερά δεξιά εναλλάξ , όπως θα αναλύσουμε παρακάτω.

Όσο αφορά τα χαρακτηριστικά των αντικειμένων για το συγκεκριμένο πρόβλημα, θα μπορούσαμε να πούμε ότι έχει δύο κύρια το βάρος και την κατάσταση, όπου η κατάσταση χωρίζεται σε δύο επιπλέον , την κατάσταση της δεξιάς και της αριστερης πλευράς. Το βάρος W είναι σαν το απλό BPP. Η κατάσταση , μας δείχνει αν η τοποθέτηση του αντικειμένου σε κάδο (ή κάδους) έχει ολοκληρωθεί. Καλό είναι σε αυτό το σημείο να θυμηθούμε ότι , κάθε αντικείμενο πρέπει να μπει στον ίδιο κάδο με το προηγούμενο του, αλλά και σε ίδιο κάδο με το επόμενο.

Οπότε η κατάσταση κάθε αντικειμένου έχει με την σειρά της δυο χαρακτηριστικά , την αριστερή πλευρά LS και την δεξιά πλευρά RS. Το πρώτο αντικείμενο X1 έχει την LS τοποθετημένη και το τελευταίο αντικείμενο Xn έχει την RS τοποθετημένη ενώ όλες οι άλλες πλευρές είναι προς τοποθέτηση εξ ορισμού τους. Η LS του X_k είναι η RS του X_{k-1} και η RS του X_k είναι η LS του X_{k+1} . Όταν η δεξιά πλευρά RS ενός αντικείμενου έχει τοποθετηθεί, τότε το αντικείμενο αυτό δεν μπορεί να πραγματοποιήσει πλέον έλεγχο , για προς εισαγωγή αντικείμενο στον δεξιό του αντικείμενο (προφανώς το αντικείμενο στα δεξιά του είναι ήδη σε κάδο και μάλιστα στον ίδιο κάδο), άλλα ούτε μπορεί να εισέλθει σε άλλο κάδο από έλεγχο που πραγματοποιείται για προς εισαγωγή αντικείμενα προχωρώντας βήματα προς τα αριστερά.

Έχοντας αυτά αποσαφηνισμένα μπορούμε να προχωρήσουμε στην παρουσίαση του αλγορίθμου. Ξεκινώντας από το μεγαλύτερο, από άποψη βάρους αντικείμενο , και τον πρώτο κάδο, ελέγχουμε για εισαγωγή το πρώτο δεξιά αντικείμενο και στην συνέχεια το πρώτο από τα αριστερά, δηλαδή αν το $X_κ$ έχει το maxW ελέγχουμε για εισαγωγή πρώτα το $X_{κ+1}$ και στην συνέχεια το $X_{κ-1}$.Η σειρά πρώτα δεξιά και στην συνέχεια αριστερά επιλέχθηκε εντελώς αυθαίρετα. Αν το $X_{κ+1}$ μπορεί να μπει στον κάδο η RS του $X_κ$ και η LS του $X_{κ+1}$ αλλάζουν την κατάσταση τους σε τοποθετημένες και ο έλεγχος προς τα δεξιά περνά στο $X_{κ+1}$. Στην συνέχεια ελέγχει το στοιχείο αριστερά του $X_κ$ δηλαδή το $X_{κ-1}$.Αν μπορεί να μπει στον κάδο η LS του Χκ και η RS του Χκ-1 αλλάζουν κατάσταση σε τοποθετημένες και ο έλεγχος πρως τα αριστερά περνά στο Χκ-1. Αυτό συνεχίζεται έως ότου δεν μπορούμε να τοποθετήσουμε το επόμενο αντικείμενο. Έστω ότι δεν μπορούμε να τοποθετήσουμε το επόμενο αντικείμενο προς τα δεξιά , σταματάμε τον έλεγχο προς τα δεξιά κ συνεχίζουμε προς τα αριστερά και μόνο. Όταν σταματήσει ο έλεγχος και προς τα αριστερά, έχουμε ένα ολοκληρωμένο κάδο. Δημιουργούμε τον επόμενο κάδο διαλέγοντας το μεγαλύτερο διαθέσιμο από άποψη W αντικείμενο το τοποθετούμε στον κάδο και επαναλαμβάνουμε το ίδιο, έως ότου να μην υπάρχει διαθέσιμο αντικείμενο. Αν το αντικείμενο που διαλέξαμε είναι διαθέσιμο αλλά η μια του πλευρά είναι τοποθετημένη πχ. η LS τότε θα πραγματοποιήσει έλεγχο μόνο ως προς την άλλη πλευρά , την δεξιά για το παράδειγμά μας.

Για να κρίνουμε αν ένα αντικείμενο μπορεί να μπει σε ένα κάδο όπως πρέπει να είναι πλέων αντιληπτό, δεν εξαρτάται αποκλειστικά από το W, αλλά πρώτα από όλα από το αν είναι διαθέσιμη για έλεγχο η κάθε πλευρά του. Για την LS του προς εισαγωγή αντικείμενου ,ο έλεγχος από τα αριστερά , θα είναι έλεγχος δεξιάς πλευράς σε σχέση με το αντικείμενο που την πραγματοποιεί (άλλα κ το πρώτο αντικείμενο που μπήκε στον κάδο) και αντίστροφα για την RS. Αν το αντικείμενο μπορεί να ελεγχθεί για εισαγωγή , θα μπει η δεν θα μπει ανάλογα με το βάρος του W. Αν μπει θα αυξήσει στην πληρότητα του κάδου κατά W και το αντικείμενο αυτό θα αποτελέσει το επόμενο σημείο ελέγχου. Αν δεν μπει, σταματάει ο έλεγχος προς αυτή την πλευρά και συνεχίζει αποκλειστικά προς την άλλη, το τελευταίο αντικείμενο που μπήκε προς αυτή την πλευρά είναι η μια άκρη του κάδου.

Αν έχουν προκύψει κάδοι συμπληρωμένοι κατά λιγότερο από το μισό , τους αντιμετωπίζουμε σαν ενιαία αντικείμενα και εφαρμόζουμε σε αυτά τον αλγόριθμο FFD.

Θα μπορούσαμε να συνοψίσουμε τα παραπάνω στα έξεις βήματα

ΒΗΜΑ 1ο

Ταξινομούμε τα αντικείμενα σε λίστα με φθίνουσα σειρά με βάση το βάρος W κάθε αντικειμένου

ΒΗΜΑ 2ο

Περνούμε ένα νέο κάδο και βάζουμε το μέγιστο στοιχείο της λίστας Xi που έχει τουλάχιστον μια πλευρά μην τοποθετημένη σε κάδο

ΒΗΜΑ 3ο

Προσπαθούμε να βάλουμε τα διαθέσιμα δεξιά, αριστερά αντικείμενα στον κάδο κινούμενοι δεξιά, αριστερά εναλλάξ

α) για κάθε αντικείμενο δεξιά (X_{i+1}) που μπορεί να μπει στον κάδο (δεν είναι ήδη τοποθετημένη η αριστερή του πλευρά κ το W του είναι μικρότερο της υπολειπόμενης χωρητικότητας του κάδου) το τοποθετούμε (θέτουμε την αριστερή πλευρά του (X_{i+1}) ως

τοποθετημένη όπως κ την δεξιά του (Xi) και στην επόμενη φορά ελέγχουμε το δεξιά αυτού (X_{i+2}). Όταν κάποιο στοιχείο δεν μπορεί να μπει σταματάμε τον έλεγχο προς τα δεξιά.

β) για κάθε αντικείμενο αριστερά (X_{i-1}) που μπορεί να μπει στον κάδο (δεν είναι ήδη τοποθετημένη η δεξιά του πλευρά κ το W του είναι μικρότερο της υπολειπόμενης χωρητικότητας του κάδου) το τοποθετούμε (θέτουμε την δεξιά πλευρά του (X_{i-1}) ως τοποθετημένη όπως κ την αριστερή του (Xi)) και στην επόμενη φορά ελέγχουμε το αριστερά αυτού (X_{i-2}). Όταν κάποιο στοιχείο δεν μπορεί να μπει σταματάμε τον έλεγχο προς τα αριστερά.

ΒΗΜΑ 4ο

Όταν σταματήσει ο έλεγχος αριστερά κ δεξιά ο κάδος έχει ολοκληρωθεί και ξεκινάμε το γέμισμα νέου κάδου επαναλαμβάνοντας τα βήματα 2,3 έως ότου να μην υπάρχει άλλο αντικείμενο στην λίστα. Αν δεν υπάρχει αντικείμενο στην λίστα οι κάδοι που δημιουργήθηκαν είναι το αποτέλεσμα της λύσης μας.

ΒΗΜΑ 5ο

Αν έχουν προκύψει κάδοι συμπληρωμένοι κατά λιγότερο από το μισό , τους αντιμετωπίζουμε σαν ενιαία αντικείμενα και εφαρμόζουμε σε αυτά τον αλγόριθμο FFD.

Παράδειγμα 2.6.2

Να κατανεμηθούν τα 2-3-5-4-2 σε κάδους χωρητικότητας C=10 με τον αλγόριθμο maxRL

(2]-[3]-[5]-[4]-[2)	bins [5]
(2]-[3]-[5]-[4]-[2)	bins [5,4]
(2]-[3]-[5]-[4]-[2)	bins [5,4], [5]
(2]-[3]-[5]-[4]-[2)	bins [5,4], [5,3]
(2]-[3]-[5]-[4]-[2)	bins [5,4], [5,3,2]
(2]-[3]-[5]-[4]-[2)	bins [5,4], [5,3,2], [4]
(2]-[3]-[5]-[4]-[2)	bins [5,4], [5,3,2], [4,2]

Παράδειγμα 2.6.3

Να κατανεμηθούν τα 4-4-2-1-5-1-1-3 σε κάδους χωρητικότητας C=10 με τον αλγόριθμο maxRL.

(4]-[4]-[2]-[1]-[5]-[1]-[1]-[3)	bin[5]
(4]-[4]-[2]-[1]-[5]-[1]-[1]-[3)	bin[5,1]
(4]-[4]-[2]-[1]-[5]-[1]-[1]-[3)	bin [5,1,1]

(4]-[4]-[2]-[1]-[5]-[1]-[1]-[3)	bin [5,1,1,1]
(4]-[4]-[2]-[1]-[5]-[1]-[1]-[3)	bin [5,1,1,1,2]
(4]-[4]-[2]-[1]-[5]-[1]-[1]-[3)	bin [5,1,1,1,2] bin[4]
(4]-[4]-[2]-[1]-[5]-[1]-[1]-[3)	bin [5,1,1,1,2] bin[4,2]
(4]-[4]-[2]-[1]-[5]-[1]-[1]-[3)	bin [5,1,1,1,2] bin[4,2,4]
(4]-[4]-[2]-[1]-[5]-[1]-[1]-[3)	bin [5,1,1,1,2] bin[4,2,4] bin[3]
(4]-[4]-[2]-[1]-[5]-[1]-[1]-[3)	bin [5,1,1,1,2] bin[4,2,4] bin[3,1]

2.6.5 Αλγόριθμος ζευγών

Ένας άλλος αλγόριθμος που μπορεί να χρησιμοποιηθεί για την λύση του προβλήματος είναι να δούμε τα διαδοχικά αντικείμενα σαν ένα αντικείμενο με βάρος το άθροισμα των βαρών τους και εν συνεχεία να εφαρμόσουμε στο αποτέλεσμα τους έναν από τους αλγορίθμους του κλασικού bin packing κατά βούληση. Πακετάροντας όλα τα διαδοχικά x_n , x_{n+1} παίρνοντας ουσιαστικά το βάρος της πλευράς n που ενώνει τα x_n, x_{n+1}. Το αρνητικό της λύσης είναι ότι όλα τα βάρη πλην του βάρους του πρώτου και το τελευταίου αντικείμενου θα επαναληφθούν. Το θετικό είναι ότι από την στιγμή που θα έχουμε τα βάρη προς τοποθέτηση έως ζευγάρια μπορούμε να χρησιμοποιήσουμε ήδη γνωστούς αλγορίθμους για την λύση του προβλήματος.

ΒΉΜΑ 1ο

Για κάθε ζεύγος διαδοχικών αριθμών παίρνουμε το συνολικό τους βάρος και το βλέπουμε έως ένα αντικείμενο. Έστω $x_1 + x_2$ το πρώτο αντικείμενο με $W1 = wx_1 + x_2$, $W2 = wx_2 + x_3$,…, $Wn\text{-}1 = wx_{n-1} + wx_n$

ΒΗΜΑ 2ο

Λύνουμε για τα βάρη W1,…,Wn-1 το κλασικό bin packing χρησιμοποιώντας κατά βούληση έναν από τους γνωστούς αλγόριθμους.

Όπως είναι εύκολο να παρατηρηθεί το $\sum_{i=1}^{n-1} Wi = 2\sum_{i=1}^{n} wi$ -(w1+wn)

Άρα $\sum_{i=1}^{n-1} Wi < \sum_{i=1}^{n} wi$ που σημαίνει ότι $\sum_{i=1}^{n-1} Wi < 2OPT$ για το κλασικό BPP, ανάλογα με τον αλγόριθμο που θα χρησιμοποιήσουμε για την λύση , μπορούμε να έχουμε το worst case για το κλασικό BPP επί 2.

Παράδειγμα 2.5.4

Να κατανεμηθούν τα 4-4-2-1-5-1-1-3 σε κάδους χωρητηκότητας C=10 με τον αλγόριθμο ζευγών.

Παίρνουμε για αρχή τα αθροίσματα των πλευρών.

w1=x1+x2 =8, w2= x2+x3=6,w3= x3+x4=3, w4=x4+x5=6, w5=x5+x6=2, w7=x7+x8

Άρα έχουμε να λύσουμε το κλασικό Bin Packing για τα βάρη 8,6,3,6,6,3,2

Λύνοντας με BFD έχουμε [8,2] ,[6,3],[6,3] [6] άρα ο αλγόριθμος μας θα χρειαστεί 4 κάδους (έναντι 3 για τους άλλους αλγόριθμους)

2.6.6 Αλγόριθμος ζευγών-NF Hybrid

Ένας ακόμα τρόπος προσεγγιστικής επίλυσης του προβλήματος είναι να χειριστούμε διαφορετικά τα ζευγάρια που προκύπτουν. Ξεκινώντας το ταίριασμα των αντικειμένων για κάθε x_n , x_{n+1} , θα τα μεταχειριστούμε διαφορετικά ανάλογα με το άθροισμα των βαρών τους. Αν $wx_n + wx_{n+1} \geq c/2$ θα τα τοποθετήσουμε σε ένα κάδο και θα συνεχίσουμε με Next-Fit αλγόριθμο προσπαθώντας να τοποθετήσουμε όσο το δυνατόν περισσότερα αντικείμενα, προσπαθώντας να φτιάξουμε ένα όσο το δυνατών πιο συμπληρωμένο κάδο, όταν φτάσουμε κάποιο αντικείμενο που δεν χωρά να εισέλθει ξεκινάμε νέο έλγχο μεταξύ του τελευταίου στοιχείου που εισήλθε κ του επόμενου του. Στην περίπτωση που για κάποιο $wx_n + wx_{n+1} < c/2$ πέρνουμε το ζεύγος και το χρησιμοποιούμε σαν ενιαίο στοιχείο, το νέο στοιχείο αυτό, έστω Wn, θα αντιπροσωπεύει το άθροισμα των βαρών. Μόλις περάσουμε από όλα τα αντικείμενα τα έχουμε ως αποτέλεσμα δύο ήδη στοιχείων από την μια κάδους συμπληρωμένους με Next-Fit και από την άλλη στοιχεία μεγέθους < c/2. Θα ολοκληρώσουμε την συμπλήρωση τον κάδων με τον αλγόριθμο BFD.

Β΄ΗΜΑ 1ο

Για κάθε ζεύγος διαδοχικών αριθμών που ελέγχουμε υπολογίζουμε το συνολικό τους βάρος ξεκινώντας από το x_1 +x .

Β΄ΗΜΑ 2ο

Αν το άθροισμα που προκύπτει wxn + wxn+1 < c/2 τότε αντιμετωπίζουμε τα xn , xn+1 σαν ένα νέο αντικείμενο με βάρος το άθροισμα το βαρών τους κ προχωράμε τον έλεγχο στο xn+1 και στο διαδοχικό του.

Β΄ΗΜΑ 3ο

Αν το άθροισμα που προκύπτει $wx_n + wx_{n+1} \geq c/2$ τότε τα τοποθετούμε σε ένα κάδο και τον γεμίζουμε με NF αλγοριθμο. Μόλις σταματήσει ο αλγόριθμος περνάμε τον έλεγχο στο τελευταίο στοιχείο που μπήκε στον κάδο και στον διαδοχικό του

Β΄ΗΜΑ 4ο

Τοποθετούμε τα νέα στοιχεία που έχουν δημιουργηθεί με βάρος <c/2 στους ήδη υπάρχοντες κάδους και στην συνέχεια μεταξύ τους χρησιμοποιώντας αλγόριθμο BFD

Παράδειγμα 2.5.5

Να κατανεμηθούν τα 4-4-2-1-5-1-1-3 σε κάδους χωρητηκότητας C=10 με τον ανωτέρω αλγοριθμο .

$x_1+x_2=8$ ακολουθούμε NF [4,4,2]

$x_3+x_4=3$ το αφήνουμε ώς νεο αντικείμενο με 'x_3 με w=3

$x_4+x_5=6$ ακολουθούμε NF [1,5,1,1]

$x_7+x_8=4$ ο αφήνουμε ώς νεο αντικείμενο με 'x_7 με w=4

Φτάσαμε στο σημείο που έχουμε τους κάδους [4,4,2],[1,5,1,1] και τα στοιχεία 'x_3 με w=3 και 'x_7 με w=4 . Εφαρμόζουμε BFD και περνούμε [4,4,2],[1,5,1,1], [4,3]

3. ΣΥΜΠΕΡΑΣΜΑΤΑ

Το BPP παρότι είναι είναι ένα πρόβλημα μελετημένο διεξοδικά παραμένει σύγχρονο, καθώς οι διάφορες εκδοχές του όπως και οι εφαρμογές που αυτές έχουν το καθιστούν ενδιαφέρον. Ιδιαίτερα οι εφαρμογές του στον χώρο της βιομηχανίας και της πληροφορικής , αποτελούν αιτία περαιταίρω μελέτης του και αναζωπύρωσης του ενδιαφέροντος. Τόσο στους απλοϊκούς αλγόριθμους, όσο και στους exact το πεδίο μελέτης παραμένει ανοιχτό. Ενώ ιδιαίτερο ενδιαφέρον παρουσιάζει η μελέτη του Bin Packing Problem παράλληλα με άλλα προβλήματα όπως το MapReduce καθώς και πως μπορούμε να αξιοποιήσουμε συνδυαστικά τους αλγόριθμους τον προβλημάτων ώστε να οδηγηθούμε σε καλύτερα αποτελέσματα. Όσο αφορά τη μελέτη του BPP με τοποθέτηση διαδοχικών στοιχείων σε κοινό κάδο σε μονοπάτι η πρόκλησή παραμένει ανοιχτή στο να βρούμε αποδοτικούς αλγόριθμους , αλλά και την επέκταση τους σε άλλες δομές πέρα από το μονοπάτι όπως τα δέντρα και οι κλίκες.

ΑΝΑΦΟΡΕΣ

[1]F. Clautiaux, A. Jouglet, J. E. Hayek, A new lower bound for the non-oriented twodimensional bin-packing problem, Operations Research Letters, Vol. 35, No. 3, 365-373, 2007.

[2] A. Lodi, S. Martello, M. Monaci, Two-dimensional packing problems: A survey, European Journal of Operational Research, Vol. 141, No. 2, 241-252, 2002.

[3]D. Pisinger, M. Sigurd, The two-dimensional bin packing problem with variable binsizes and costs, Discrete Optimization, Vol. 2, No. 2, 154-167, 2005.

[4]S. Martello, D. Pisinger, D. Vigo, The three-dimensional bin packing problem, Operations Research, Vol. 48, 256-267, 2000.

[5]D. S. Liu, K. C. Tan, S. Y. Huang, C. K. Goh, W. K. Ho, On solving multiobjectivebin packing problems using evolutionary particle swarm optimization, European Journal of Operational Research, Vol. 190, No. 2, 357-382, 2008.

[6],L. Babel, B. Chen, H. Kellerer, V. Kotov, Algorithms for on-line bin-packing problems with cardinality constraints, Discrete Applied Mathematics, Vol. 143, Nos 1-3, 238-251, 2004.

[7] E. G. Coffman, M. R. Garey, D. S. Johnson, Dynamic bin packing, SIAM Journal of Computation, Vol. 12, 227-258, 1983.

[8]Epstein, M. Levy, Dynamic multi-dimensional bin packing, Journal of Discrete Algorithms, Vol. 8, No. 4, 356-372, 2010.

[9] J. Puchinger, G. R. Raidl, Models and algorithms for three-stage two-dimensional bin packing, European Journal of Operational Research, Vol. 183, No. 3, 1304-1327, 2007.

[10] Bang-Jensen, R. Larsen, Efficient algorithms for real-life instances of the variable size bin packing problem, Computers & Operations Research, Vol. 39, No. 11, 2848-2857, 2012.

[11] Boyar, L. M. Favrholdt, A new variable-sized bin packing problem, Journal of Scheduling, Vol. 15, No. 3, 273-287, 2012.

[12] J. Kang, S. Park, Algorithms for the variable sized bin packing problem, European Journal of Operational Research, Vol. 147, 365-372, 2003.

[13] D. S. Johnson, Fast algorithms for bin packing, Journal of Computer and System Sciences, Vol. 8, 272-314.

[14] F. T. S. Chan, K. C. Au, L. Y. Chan, T. L. Lau, Using genetic algorithms to solve quality-related bin packing problem, Robotics and Computer-Integrated Manufacturing.

[15] M. Gendreau, G. Laporte, F. Semet, Heuristics and lower bounds for the bin packing problem with conflicts, Computers & Operations Research,Vol. 31, No. 3, 347-358, 2004.

[16] A. Stawowy, Evolutionary based heuristic for bin packing problem, Computers & Industrial Engineering, Vol. 55, No. 2, 465-474, 2008.

[17](Jr. E. Co®man, M. Garey, and D. Johnson. In D. Hochbaum, editor, Approximation Algorithms for NP-Hard Problems, pages 46{93. PWS Publishing Company, Massachusetts.).

[18]S. Martello and P. Toth. Knapsack problems: algorithms and computer implementations. Chichester: John Wiley & Sons, 1990.

[19] Silvano Martello and Paolo Toth. Lower bounds and reduction procedures for the bin packing problem. Discrete Applied Mathematics, 28(1):59–70, 1990).

[20] [1971 S. Eilon and N. Christofides.The loading problem. Management Science, 17(5):259–268, 1971.

[21] R.E. Korf. A new algorithm for optimal bin packing. In Proceedings of the National conference on Artificial Intelligence, pages 731–736, 2002.

[22]Richard E. Korf. An improved algorithm for optimal bin packing. In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03) Acapulco, Mexico, pages 1252–1258, 200.

[23] F. Afrati, S. Dolev, E. Korach, S. Sharma, and J. D. Ullman Assignment Problems of Different-Sized Inputs in MapReduce.

[24] X. Han, C. Peng, D. Ye, D. Zhang, Y. Lan, Dynamic bin packing with unit fraction items revisited, Information Processing Letters, Vol. 110, No. 23, 1049-1054, 2010.