Πανεπιστήμιο
Ιωαννίνων

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΜΑΘΗΜΑ:ΑΛΓΟΡΙΘΜΟΙ & ΠΡΟΧΩΡΗΜΕΝΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΘΕΜΑ : ΤΕΧΝΙΚΗ ΑΝΑΦΟΡΑ ΓΙΑ 0-1 KNAPSACK PROBLEM

ΕΚΠΟΝΗΣΗ : ΖΑΧΑΡΗΣ ΑΡΙΣΤΟΤΕΛΗΣ

Α.Μ.: 52

ΗΜ/ΝΙΑ: 02/01/2019

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΓΚΟΓΚΟΣ ΧΡΗΣΤΟΣ

Περιεχόμενα

Περίληψη

Στην εργασία μας θα ασχοληθούμε με τον 0-1 Knapsack Problem. Στην αρχή θα κάνουμε μία εισαγωγή για τον κάθε αλγόριθμο ξεχωριστά να αναφερθούμε στην πολυπλοκότητα του κάθε ενός από τους 6 και που υστερούν ή υπερτερούν μεταξύ τους και θα βγάλουμε συμπεράσματα για 6 αλγόριθμους για το ποιος από αυτούς προσφέρει την βέλτιστη λύση μιας και ο χρόνος εκτέλεσης τους θα είναι τα 10 sec. Μετά το πέρας του προαναφερθέντος χρόνου οι αλγόριθμοι θα επιστρέφουν τις τιμές που έχουν βρει μέχρι τότε για τα εκάστοτε προβλήματα. Τέλος, θα παρουσιαστούν πίνακες και γραφήματα για την συμπεριφορά που δίνουν οι αλγόριθμοι.

Εισαγωγή

Στην οικογένεια των Knapsack Problems δίδεται ένα σύνολο από αντικείμενα και ζητείται να επιλεγούν αυτά που θα μεγιστοποιήσουν το αντίστοιχο κέρδος, χωρίς όμως να παραβιάζεται η χωρητικότητα του ή των σακιδίων αντίστοιχα. Υπάρχουν διαφορετικά είδη Knapsack Problems ανάλογα με τον τρόπο κατανομής των αντικειμένων και των σακιδίων. Όπως προαναφέρθηκε θα ασχοληθούμε με το 0-1 Knapsack Problem, όπου κάθε αντικείμενο επιλέγεται το πολύ μία φορά. Θα αναφερθούμε στις τεχνικές άπληστη μέθοδος(greedy approach), εξαντλητική απαρίθμηση συνδυασμών **(**brute force full Enumeration) , διακλάδωση και φραγή **(**branch and bound) του δυναμικού προγραμματισμού (Dynamic Programming), του ακέραιου προγραμματισμού **(**Integer Programming) , και ενός εξειδικευμένου επιλυτή μέσω του OR-TOOLS που βασίζεται στον δυναμικό προγραμματισμό και πως αυτές μπορούν να χρησιμοποιηθούν για να λύσουν το 0-1 Knapsack Problem.

1.Αλγόριθμοι

1.1 Άπληστη μέθοδος (Greedy Approach)

Στην άπληστη μέθοδο (greedy approach) θα υπολογίζεται για κάθε αντικείμενο ο λόγος αξία προς βάρος και θα δίνεται προτεραιότητα στην εισαγωγή των αντικειμένων με τις μεγαλύτερες τιμές και μέχρι να μην μπορεί να προστεθεί άλλο αντικείμενο στο σακίδιο.

1.2 Εξαντλητική απαρίθμηση συνδυασμών (Brute Force Approach)

Ένας απλός τρόπος είναι τη στιγμή που έχουμε να επιλέξουμε ανάμεσα σε n αντικείμενα για να γεμίσουμε το σακίδιο, να πάρουμε όλους τους πιθανούς συνδυασμούς των αντικειμένων, δηλαδή τους 2^n συνδυασμούς, να τους εξετάσουμε προκειμένου να βρούμε την βέλτιστη λύση η οποία οδηγεί στο μεγαλύτερο δυνατό κέρδος υπό την προϋπόθεση ότι δεν ξεπερνά την συγκεκριμένη χωρητικότητα του σακιδίου. Η τεχνική αυτή έχει χρόνο εκτέλεσης O($n2^n$).

1.3 Διακλάδωση και φραγή (Branch and Bound)

Η μέθοδος διακλάδωσης και φραγής (branch and bound) αποτελεί βελτίωση της μεθόδου πλήρους απαρίθμησης. Αρχικά τα αντικείμενα διατάσσονται σε φθίνουσα σειρά αξίας προς βάρος. Στη συνέχεια δημιουργείται ένα δυαδικό δένδρο μερικών λύσεων. Σε κάθε κόμβο του δένδρου λαμβάνεται η απόφαση επιλογής ενός αντικειμένου η οποία οδηγεί στο αριστερό υποδένδρο ενώ η μη επιλογή του οδηγεί

στο δεξιό υποδένδρο. Επιπλέον σε κάθε κόμβο καταγράφεται το συνολικό βάρος και η συνολική αξία των αντικειμένων του. Η βασική ιδέα είναι ότι κάποιοι κλάδοι του δένδρου δεν έχει νόημα να δημιουργηθούν καθώς δεν πρόκειται να οδηγήσουν σε καλύτερη λύση από κάποια που έχει ήδη βρεθεί. Αυτό γίνεται με την καταγραφή της καλύτερης τιμής που έχει εντοπιστεί ανά πάσα στιγμή και τον υπολογισμό ενός φράγματος καλύτερης περίπτωσης (άνω φράγμα) για κόμβους που πρόκειται να αναπτυχθούν προσθέτοντας επιπλέον αντικείμενα στα ήδη υπάρχοντα. Ένας απλός τρόπος υπολογισμού του άνω φράγματος είναι να προστεθεί στην τρέχουσα αξία ενός κόμβου (δηλαδή στη συνολική αξία των αντικειμένων του κόμβου), το γινόμενο της χωρητικότητας του σακιδίου που απομένει ελεύθερο με τον καλύτερο λόγο αξίας προς βάρος των αντικειμένων που απομένουν προς επιλογή. Αν η τιμή του άνω φράγματος είναι μικρότερη από την καλύτερη τιμή που έχουμε καταγράψει τότε δεν έχει νόημα να αναπτύξουμε περαιτέρω τον κόμβο, οπότε γίνεται κλάδεμα του δένδρου. Επομένως ο χρόνος εκτέλεσης του είναι $O(2^{n-1} -1)$.

1.4 Δυναμικός προγραμματισμός (Dynamic Programming)

Ο δυναμικός προγραμματισμός αποτελεί ισχυρό αλγοριθμικό πρότυπο καθώς εδώ ένα πρόβλημα μπορεί να επιλυθεί με τον εντοπισμό μιας συλλογής υπο-προβλημάτων και αντιμετωπίζοντας αυτά ένα προς ένα, ξεκινώντας από τα μικρότερα, χρησιμοποιούνται οι απαντήσεις τους για λυθούν τα μεγαλύτερα, μέχρι το σύνολο των προβλημάτων να λυθεί.
Ο Dynamic Programming είναι μια μέθοδος που χρησιμοποιείται για την επίλυση προβλημάτων βελτιστοποίησης. Η βασική του ιδέα είναι να υπολογίσουμε μια φορά τις λύσεις μικρότερων προβλημάτων του αρχικού μας προβλήματος (subsub-problems), να τις αποθηκεύσουμε σε έναν πίνακα έτσι ώστε να μπορούμε να τις επαναχρησιμοποιήσουμε αργότερα, όταν χρειαστεί. Προσπαθούμε ουσιαστικά να πάρουμε περισσότερο χώρο για να γλυτώσουμε χρόνο. Ο χρόνος εκτέλεσης τώρα είναι O(n*W).

1.5 Ακέραιος προγραμματισμός (Integer Programming)

Ο ακέραιος προγραμματισμός (Integer Programming) αποτελεί έναν γενικό τρόπο επίλυσης προβλημάτων που είναι δυνατόν να μοντελοποιηθούν με συγκεκριμένο τρόπο. Στηρίζεται στον αλγόριθμο επαναληπτικής βελτίωσης simplex και είναι σε θέση να εντοπίζει λύσεις ακόμα και εάν υπάρχουν περιορισμοί ακεραιότητας για τις τιμές των μεταβλητών απόφασης.

2.Υλοποίηση

Για την υλοποίηση της εργασίας δημιουργήσαμε 320 στιγμιότυπα με τον generator του Pisinger. Για κάθε συνδυασμό των ακόλουθων παραμέτρων δημιουργήθηκαν 5 υπό-προβλήματα για n={10,50,100,500}, r={50,100,500,1000} και type={1,2,3,4}, δηλαδή σύνολο $5 \times 4 \times 4 \times 4 = 320$ στιγμιότυπα. Στην συνέχεια, θα παρουσιαστούν κάποια γραφήματα και κάποιοι πίνακες που θα φαίνονται ξεκάθαρα αποτελέσματα για το ποιοι είναι πιο αξιόπιστοι αλγόριθμοι και δίνει πάντα λύση και μάλιστα την βέλτιστη στο χρονικό διάστημα των 10 sec.

3.Παρουσίαση αποτελεσμάτων με πίνακες και γραφήματα

Algorithms	Items included	Total Profit	Max Profit
Greedy Approach	4,5,9,10	159	159
Brute Force	4,5,9,10	159	159
Branch and Bound	4,5,9,10	159	159
Dynamic Programming	4,5,9,10	159	159
Dynamic Programming OR-TOOLS	4,5,9,10	159	159
Integer Programming OR-TOOLS	4,5,9,10	159	159

Πίνακας 1: Απεικόνιση μέγιστης αξίας και επιλεχθέντων αντικειμένων των 6 αλγορίθμων για το πρόβλημα 10_50_1_1_5

Algorithms	Items included	Total Profit	Max Profit
Greedy Approach	4,5,9,10,11,12,16,17,19,24,30,34,36,37,50	581	581
Brute Force	4,5,9,10,11,16,19,36,37,41,42,43,48	**486**	581
Branch and Bound	4,5,9,10,11,12,16,17,19,24,30,34,36,37,50	581	581
Dynamic Programming	4,5,9,10,11,12,16,17,19,24,30,34,36,37,50	581	581
Dynamic Programming OR-TOOLS	4,5,9,10,11,12,16,17,19,24,30,34,36,37,50	581	581
Integer Programming OR-TOOLS	4,5,9,10,11,12,16,17,19,24,30,34,36,37,50	581	581

Πίνακας 2: Απεικόνιση μέγιστης αξίας και επιλεχθέντων αντικειμένων των 6 αλγορίθμων για το πρόβλημα με 50_50_1_1_5

Algorithms	Total Weight	Total Profit	Time
Greedy Approach	941	2249	0
Brute Force	939	**1496**	10
Branch and Bound	**484**	**1599**	10
Dynamic Programming	941	2299	0
Dynamic Programming OR-TOOLS	941	2299	0
Integer Programming OR-TOOLS	941	2299	0

Πίνακας 3: Απεικόνιση συνολικής αξίας, συνολικού βάρους και χρόνου εκτέλεσης των 6 αλγορίθμων για το πρόβλημα 100_100_1_1_5

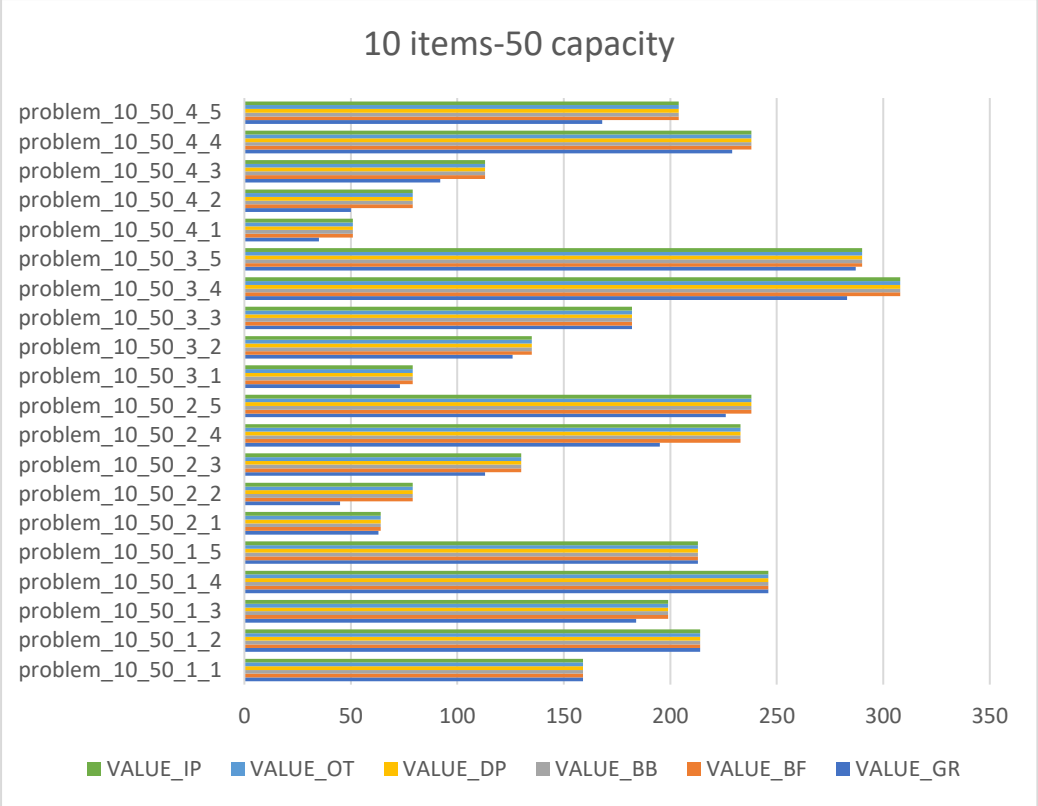
Algorithms	Total Weight	Total Profit	Time
Greedy Approach	21577	58126	0
Brute Force	21653	**23624**	10
Branch and Bound	**319**	**6784**	10
Dynamic Programming	21655	58221	0
Dynamic Programming OR-TOOLS	21655	58221	0
Integer Programming OR-TOOLS	21655	58221	0

Πίνακας 4: Απεικόνιση συνολικής αξίας, συνολικού βάρους και χρόνου εκτέλεσης των 6 αλγορίθμων για το πρόβλημα 500_500_1_1_5

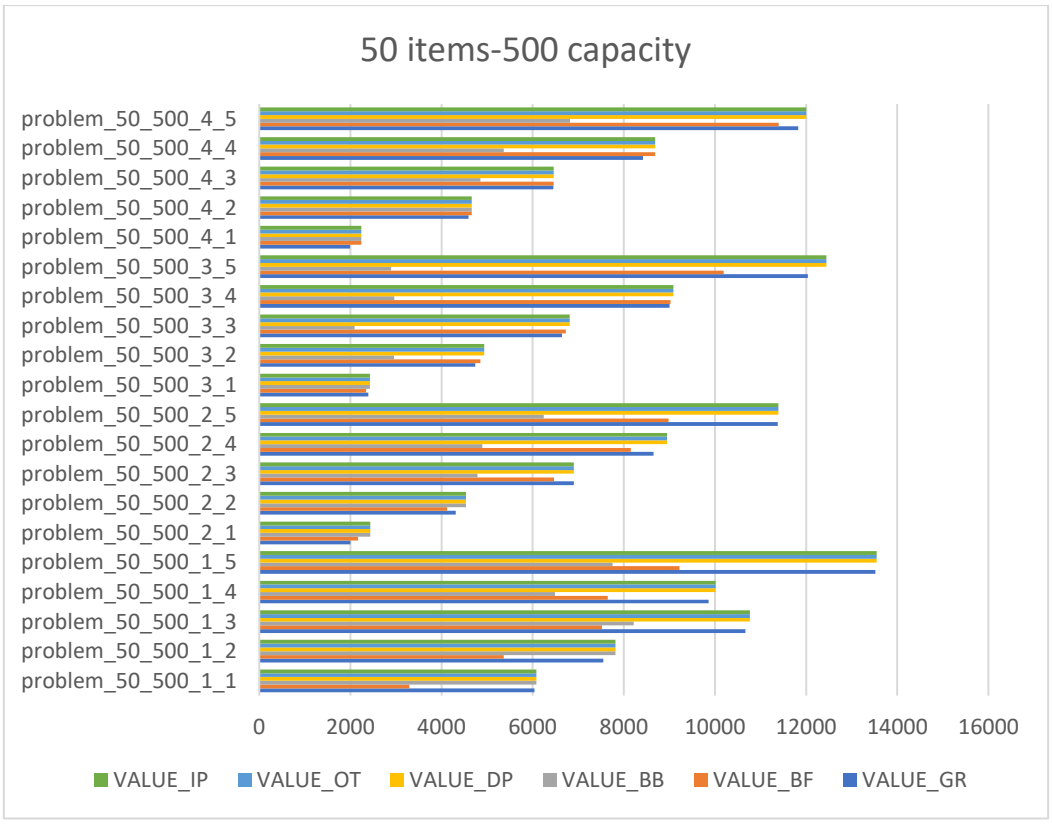
Algorithms	Total Weight	Total Profit	Time
Greedy Approach	42500	114437	1
Brute Force	42753	**46174**	10
Branch and Bound	**668**	**15131**	10
Dynamic Programming	42822	114892	5
Dynamic Programming OR-TOOLS	42822	114892	0
Integer Programming OR-TOOLS	42822	114892	0

Πίνακας 5: Απεικόνιση συνολικής αξίας, συνολικού βάρους και χρόνου εκτέλεσης των 6 αλγορίθμων για το πρόβλημα 500_1000_1_1_5

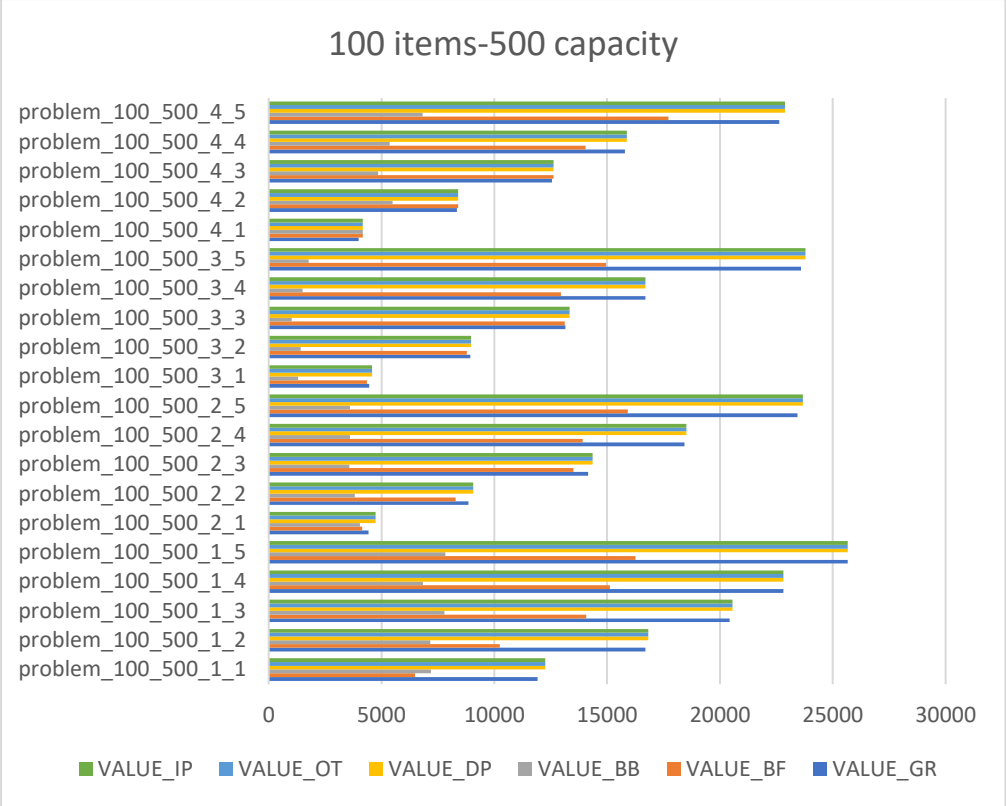
Σύμφωνα με τους ενδεικτικούς παραπάνω πίνακες βλέπουμε ότι οι αλγόριθμοι Brute Force και Branch and Bound υστερούν πάρα πολύ όσο μεγαλώνουν τα προβλήματα η χωρητικότητα και η αξία του κάθε αντικειμένου. Για αυτό και η εργασία υλοποιήθηκε για μέγιστο χρόνο εκτέλεσης των εντολών κάθε αλγορίθμου τα 10sec γιατί λόγο πολυπλοκότητας των 2 προαναφερθέντων αλγορίθμων θα υπάρξει μεγάλη καθυστέρηση για την εξαγωγή των πειραμάτων μας. Οι υπόλοιποι 4 αλγόριθμοι ακολουθούν ο ένας τον άλλο δίνοντας λύσεις αλλά οι λύσεις που δίνει ο Greedy Approach αλγόριθμος στις περισσότερες των περιπτώσεων είναι πάρα πολύ πίσω από την βέλτιστη λύση την οποία πετυχαίνουμε με τους άλλους 3 αλγορίθμους τον δυναμικό, τον ακέραιο προγραμματισμό και τον εξειδικευμένο που βασίζεται στον δυναμικό προγραμματισμό του or-tools.Στην συνέχεια θα παρουσιάσω σε κάποια ενδεικτικά γραφήματα τα αποτελέσματα που πάρθηκαν για τα 320 στιγμιότυπα που δημιουργήθηκαν και θα βγουν χρήσιμα συμπεράσματα για το ποιος υπερτερεί σε σχέση με τους άλλους αλγόριθμους και ποιος δίνει πάντα λύση και μάλιστα την βέλτιστη. Τα γραφήματα απεικονίζουν ανά 20 υπό-προβλήματα για κάθε ζευγάρι αντικειμένων και χωρητικότητας τις αξίες που επέστρεψαν σαν λύση για το κάθε πρόβλημα οι 6 αλγόριθμοι σε ραβδογράμματα.



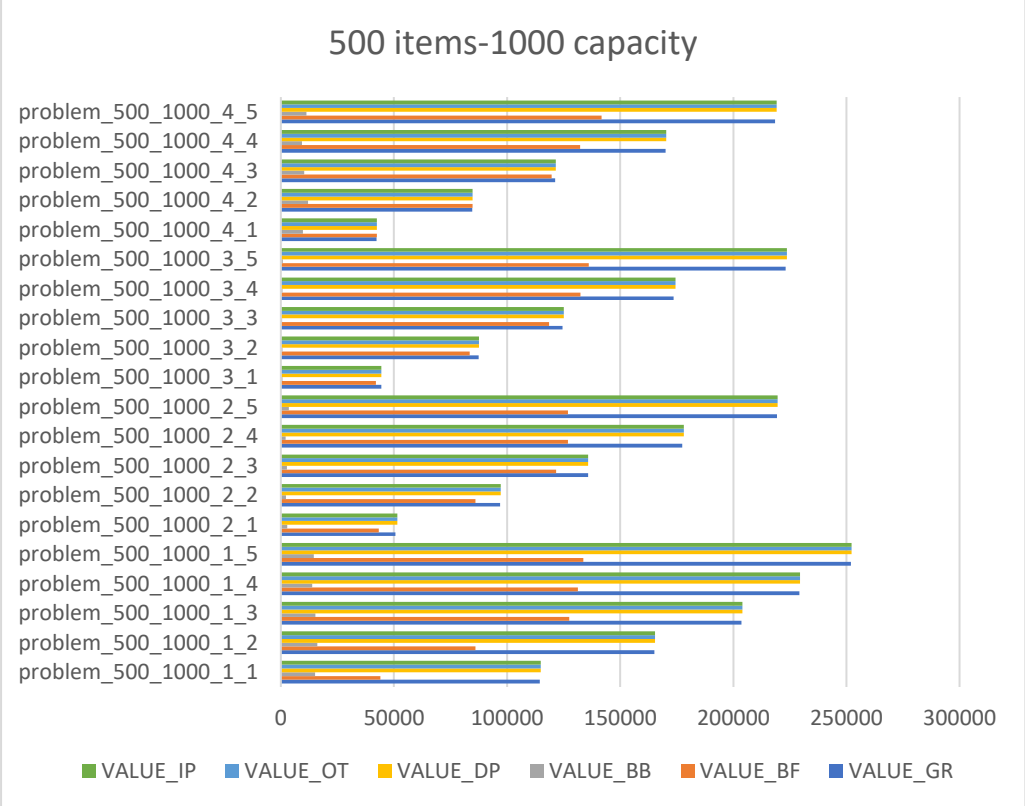
Γράφημα 1: *Στο γράφημα βλέπουμε ορισμένα αποτελέσματα για ομάδα προβλημάτων με 10 αντικείμενα. Παρατηρείται ότι ο Greedy δεν δίνει το βέλτιστο αποτέλεσμα.*



Γράφημα 2: *Με την αύξηση των αντικειμένων βλέπουμε ότι ο αλγόριθμοι Brute Force και Branch and Bound λόγω time out δεν δίνουν το βέλτιστο αποτέλεσμα και διαφοροποιούνται από τους υπόλοιπους*



Γράφημα 3: *Με την αύξηση των αντικειμένων στα 100 βλέπουμε ότι ο αλγόριθμοι Brute Force και Branch and Bound λόγω time out δεν δίνουν το βέλτιστο αποτέλεσμα και διαφοροποιούνται*



Γράφημα 4: *Για προβλήματα με 500 αντικείμενα έχουμε τα ίδια ακριβώς αποτελέσματα ενώ οι δυο δυναμικοί και ο ακέραιος δίνουν πάντα την βέλτιστη λύση*

4. Συμπεράσματα

Η συγκριτική μελέτη της άπληστης μεθόδου(greedy approach), της εξαντλητικής απαρίθμησης συνδυασμών **(**brute force full Enumeration) , της διακλάδωσης και φραγής **(**branch and bound) του δυναμικού προγραμματισμού (Dynamic Programming), δείχνει ότι ενώ οι περίπλοκοι αυτοί αλγόριθμοι είναι γνωστοί, η φύση του προβλήματος που εφαρμόζουν κάνει κάποιους από αυτούς πιο κατάλληλους από άλλους. Τις καλύτερες προσεγγίσεις για το πρόβλημα 0/1 του σακιδίου τις δίνει ο δυναμικός προγραμματισμός και οι εξειδικευκευμένοι επιλυτές του Or-tools ο ένας εκ των 2 χρησιμοποιεί τον δυναμικό προγραμματισμό και ο άλλος τον ακέραιο προγραμματισμό. Μαζί με αυτούς στο χρόνο που έχουμε ορίσει δίνει λύσεις και ο greedy και μάλιστα γρήγορες αλλά δεν είναι οι βέλτιστες. Όπως έχουμε δείξει, η επιλογή μεταξύ των παραπάνω 4 αλγορίθμων εκτός των εξειδικευμένων εξαρτάται από την ικανότητα του σακιδίου και το μέγεθος του πληθυσμού. Ωστόσο, μπορεί κανείς να αποφασίσει να επιλέξει δυναμικό προγραμματισμό σε σχέση με τους άλλους αλγόριθμους υπό οποιεσδήποτε συνθήκες, επειδή είναι εύκολο και απλό να κωδικοποιηθεί. Αντίθετα, οι γενετικοί αλγόριθμοι απαιτούν πολύ περισσότερο χρόνο όσον αφορά την κατανόηση των εννοιών του παραδείγματος και της προσπάθειας προγραμματισμού και η πολυπλοκότητα τους είναι μεγάλη εν αντιθέση με τον δυναμικό προγραμματισμό.