Ατομική Διπλωματική Εργασία

ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΔΙΑΧΕΙΡΙΣΗΣ ΠΟΡΩΝ ΣΤΟ ΥΠΟΛΟΓΙΣΤΙΚΟ ΝΕΦΟΣ ΜΕ ΧΡΗΣΗ ΤΕΧΝΙΚΩΝ ΙΚΑΝΟΠΟΙΗΣΗΣ ΠΕΡΙΟΡΙΣΜΩΝ

Μαρία Ερωτοκρίτου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΔΙΑΧΕΙΡΙΣΗΣ ΠΟΡΩΝ ΣΤΟ ΥΠΟΛΟΓΙΣΤΙΚΟ ΝΕΦΟΣ ΜΕ ΧΡΗΣΗ ΤΕΧΝΙΚΩΝ ΙΚΑΝΟΠΟΙΗΣΗΣ ΠΕΡΙΟΡΙΣΜΩΝ

Μαρία Ερωτοκρίτου

Επιβλέπων Καθηγητής
Δρ. Γιάννης Δημόπουλος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2014

Ευχαριστίες

Αρχικά, αισθάνομαι την ανάγκη να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή Δρ. Γιάννη Δημόπουλο για την συνεχή και ουσιαστική βοήθεια του προς εκπόνηση της παρούσας διπλωματικής εργασίας. Η καθοδήγηση του συνέβαλε στην υλοποίηση των επιμέρους στόχων μέσω της άρτιας ακαδημαϊκής του γνώσης και της ενδελεχούς παροχής των απαιτούμενων κατευθυντήριων οδηγιών.

Επίσης, δεν θα μπορούσα να μην ευχαριστήσω την οικογένεια μου και ιδιαίτερα τους γονείς μου, Σταύρο και Πελαγία για την αμέριστη συμπαράσταση, την θερμή ενθάρρυνση και τον ακατάπαυστο αγώνα τους προκειμένου να φέρω εις πέρας όλες τις ακαδημαϊκές μου υποχρεώσεις και ιδιαίτερα να ολοκληρώσω την παρούσα διπλωματική εργασία.

Περίληψη

Η παρούσα Διπλωματική Εργασία πραγματεύεται την επίλυση και τις μεθόδους βελτιστοποίησης στα προβλήματα κατανομής πόρων γύρω από το υπολογιστικό νέφος με χρήση τεχνικών ικανοποίησης περιορισμών. Συγκεκριμένα, σκοπός είναι η επίλυση του προβλήματος αυτόνομη και εικονική διαχείριση των πόρων για τον έλεγχο του εικονικού περιβάλλοντος με στόχο την αποσύνδεση της τροφοδότησης των πόρων (provisioning) από την δυναμική τοποθέτηση των εικονικών μηχανών στις φυσικές μηχανές(placement). Αυτή η αυτόνομη διαχείριση στοχεύει στην βελτιστοποίηση της συνάρτησης ικανοποίησης των εφαρμογών(global utility function-UGlobal) όπως επίσης και στην δυνατότητα αυτοματοποίησης δυναμικών τροφοδοτήσεων και τοποθετήσεων των εικονικών μηχανών. Τα δύο επίπεδα τα οποία θα υλοποιηθούν και θα πρέπει να διαχειριστούν είναι το επίπεδο τροφοδότησης εικονικών μηχανών(VM Provisioning) και το επίπεδο τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές(VM Placement). Το επίπεδο τροφοδότησης των εικονικών μηχανών είναι υπεύθυνο για την κατανομή της χωρητικότητας των πόρων με την μορφή εικονικές μηχανές σε εφαρμογές και το επίπεδο τοποθέτησης εικονικών μηχανών είναι υπεύθυνο για την αντιστοίχιση των εικονικών μηχανών σε φυσικές μηχανές.

Οι δύο φάσεις του προβλήματος τροφοδότησης και τοποθέτησης εικονικών μηχανών εκφράζονται ως προβλήματα ικανοποίησης περιορισμών και όπως θα παρουσιαστεί στην συνέχεια διεκπεραιώνονται από την επίλυση περιορισμών χρησιμοποιώντας τους επιλυτές Minizinc, Minisat+ και Bsolo. Αρχικά, το πρόβλημα «αυτόνομης και εικονικής διαχείρισης των πόρων» υλοποιήθηκε στον επιλυτή Minizinc ως πρόβλημα ικανοποίησης περιορισμών και ακολούθως μετατράπηκε σε ψευδοδυαδικό πρόβλημα και υλοποιήθηκε στους επιλυτές Minisat+ και Bsolo. Ο κάθε ένας από τους τρεις επιλυτές έχει διαφορετικές απαιτήσεις ως προς την μορφή των δεδομένων εισόδου και έτσι μέσω της γλώσσας προγραμματισμού C υλοποιήθηκε η απαίτηση αυτή.

Όπως αναλύεται κάτωθι, ο προγραμματισμός με περιορισμούς για την τοποθέτηση των εικονικών μηχανών μπορεί να μειώσει αποτελεσματικά τον αριθμό των φυσικών μηχανών για την επίτευξη του στόχου της μείωσης του λειτουργικού κόστους και την βελτίωση της χρήσης των πόρων.

Περιεχόμενα

Λίστα Ακρωνύμων και Συμβόλων	
Σύμβολα	
$A=(\alpha_1,\alpha_2,\alpha_i,..,\alpha_m)$	Σύνολο των εφαρμογών
$P=(p_1,p_2,p_j,..,p_q)$	Σύνολο των φυσικών μηχανών
$S=(S_1,S_2,S_k..,S_c)$, όπου $S_k=(S_k^{Cpu}, S_k^{Ram})$	Σύνολο των κατηγοριών εικονικών μηχανών , S_k^{Cpu} προσδιορίζει την ικανότητα της Cpu και S_k^{Ram} προσδιορίζει την ικανότητα μνήμης Ram) της εικονικής μηχανής k
C_j^{cpu}	Ικανότητα της φυσικής μηχανής p_j σε Cpu
C_j^{ram}	Ικανότητα της φυσικής μηχανής p_j σε Cpu και σε Ram
$N_i=(n_{i1},n_{i2},\ldots,n_{ik},n_{im})$, n_{ik} είναι ο αριθμός των εικονικών μηχανών της κατηγορίας S_k που αποδίδεται στην εφαρμογή α_i	Διάνυσμα κατανομής εικονικών μηχανών της εφαρμογής α_i
$U_i= f_i(N_i)= c1_{i1}*n_{i1} + c1_{i2}*n_{i2} + \ldots + c1_{im}*n_{im}$	Το επίπεδο των πόρων για την εφαρμογή α_i
$Cost_i= c2_{i1}*n_{i1} + c2_{i2}*n_{i2} + \ldots + c2_{im}*n_{im}$	Το λειτουργικό κόστος για την εφαρμογή α_i
$N_i^{max}=(n_{i1}^{max},n_{i2}^{max},n_{ik}^{max},n_{im}^{max})$	Άνω φράγμα για τον αριθμό των εικονικών μηχανών σε κάθε κατηγορία
T_i^{max}	Ο συνολικός αριθμός εικονικών μηχανών για κάθε εφαρμογή
$Uglobal$	Συνάρτηση Ικανοποίησης των εφαρμογών
e	Συντελεστής κόστους
a_i	Βαρύτητα κάθε εφαρμογής
$total$	Άθροισμα όλων των βαρυτήτων των εφαρμογών
$V=(vm_1,vm_2,vm_l,\ldots,vm_u)$	Σύνολο εικονικών μηχανών που απαιτούνται
$H_j=(h_{j1},h_{j2},h_{jl},\ldots,h_{ju})$	Σύνολο των εικονικών μηχανών που ανατίθενται στην φυσική μηχανή p_j
$D_j=(d_{j1},d_{j2},d_{j3},\ldots,d_{jc})$	Σύνολο των εικονικών μηχανών από κάθε κατηγορία εικονικών μηχανών που ανατίθενται στην φυσική μηχανή p_j
$U=\{u_1,u_2,\ldots,u_q\}$	Σύνολο φυσικών μηχανών και προσδιορίζει κατά πόσο μια φυσική μηχανή είναι ενεργή η όχι

$R=(r_1,r_2,r_l,..,r_u)$	ικανότητα των πόρων(cpu και ram) όλων των εικονικών μηχανών και συμβολίζεται με $r_l=(r_l^{cpu},r_l^{ram})$.
allocate$\{u_1,u_2....,u_c\}$, *όπου* u_i *είναι ο αριθμός εικονικών μηχανών για την κατηγορία i.*	Σύνολο το οποίο απαριθμεί πόσες εικονικές μηχανές απαιτούνται για κάθε κατηγορία
Ακρώνυμα	
VM-virtual machines	Εικονικές Μηχανές
PM-physical machines	Φυσικές Μηχανές
Vm Provisioning Phase	Φάση Τροφοδότησης Εικονικών Μηχανών
Vm Placement Phase	Φάση Τοποθέτησης Εικονικών Μηχανών
Cloud Computing	Υπολογιστικό Νέφος
Virtualization	Εικονοποίηση
Server Virtualization	Εικονοποίηση Διακομιστή
Storage Virtualization	Εικονοποίηση Αποθήκευσης
UGlobal-Global Utility Function	Συνάρτηση Ικανοποίησης εφαρμογών
Utility computing	Υπολογισμός χρησιμότητας
SLA	Συμφωνία μεταξύ δύο ή περισσοτέρων μερών όπου το ένα είναι ο πελάτης και οι άλλοι είναι οι πάροχοι υπηρεσιών
Qos	Μετρική ποιότητας της υπηρεσίας
Clause	Όρος
Literal	Λεκτικό

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός της Διπλωματικής

Σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη και η επίλυση προβλημάτων κατανομής πόρων με χρήση τεχνικών ικανοποίησης περιορισμών. Στόχος είναι η υλοποίηση του προβλήματος «αυτόνομη και εικονική διαχείριση των πόρων για τον έλεγχο του εικονικού περιβάλλοντος» έτσι που αποσυνδέει την τροφοδότηση των πόρων από την δυναμική τοποθέτηση των εικονικών μηχανών. Αυτή η αυτόνομη διαχείριση στοχεύει στην βελτιστοποίηση της συνάρτησης ικανοποίησης των εφαρμογών(UGlobal) όπως επίσης και στην δυνατότητα αυτοματοποίησης δυναμικών τροφοδοτήσεων και τοποθετήσεων των εικονικών μηχανών. Τα δύο επίπεδα τα οποία πρέπει να διαχειρίζονται είναι το επίπεδο τροφοδότησης εικονικών μηχανών(VM Provisioning) και το επίπεδο τοποθέτησης εικονικών μηχανών(VM Placement). Το επίπεδο τροφοδότησης εικονικών μηχανών είναι υπεύθυνο για την κατανομή της χωρητικότητας των πόρων με την μορφή εικονικές μηχανές σε εφαρμογές και το επίπεδο τοποθέτησης εικονικών μηχανών είναι υπεύθυνο για την αντιστοίχιση των εικονικών μηχανών σε φυσικές μηχανές.

Οι δύο φάσεις του προβλήματος, τροφοδότηση και τοποθέτηση εικονικών μηχανών εκφράζονται ως προβλήματα ικανοποίησης περιορισμών και όπως θα δούμε πιο κάτω εκφράζονται μέσα από περιορισμούς. Ο προγραμματισμός με περιορισμούς για τη τοποθέτηση εικονικών μηχανών μπορεί να μειώσει αποτελεσματικά τον αριθμό των

φυσικών μηχανών με στόχο τη μείωση του λειτουργικού κόστους και την βελτίωση της χρήσης των πόρων.

Μέσω της μελέτης διαφόρων άρθρων η οποία έχει προηγηθεί, αρχικά θα γίνει κατανόηση των βασικών εννοιών που θα χρησιμοποιηθούν μετέπειτα. Εν τέλει, απαιτείται να πραγματοποιηθούν πειράματα έτσι ώστε να είναι δυνατή η εξαγωγή συμπερασμών σε σχέση με τους επιλυτές που έχουν χρησιμοποιηθεί.

1.2 Δομή της Διπλωματικής

Στο κεφάλαιο 2, γίνεται σύντομη αναφορά γύρω από τα βασικά χαρακτηριστικά του υπολογιστικού νέφους και της εικονοποίησης και πως αυτά τα δύο συσχετίζονται. Στην συνέχεια, στο κεφάλαιο 3 περιγράφεται η θεωρία γύρω από τα προβλήματα ικανοποίησης περιορισμών καθώς και τα προβλήματα με ψευδοδυαδικούς περιορισμούς. Ακολούθως, στο κεφάλαιο 4 αναφέρονται οι διαφορετικές προσεγγίσεις επίλυσης του προβλήματος τοποθέτησης εικονικών μηχανών. Συγκεκριμένα, γίνεται αναφορά στις προσεγγίσεις Προγραμματισμός με ικανοποίηση περιορισμών, Πρόβλημα Τοποθέτησης σε Κάδους, Στοχαστικός Ακέραιος Προγραμματισμός και Γενετικός Αλγόριθμος. Στο κεφάλαιο 5 παρουσιάζεται λεπτομερής περιγραφή για την αυτόνομη και εικονική διαχείριση πόρων αφού σε αυτή βασίστηκε η υλοποίηση και η μελέτη της παρούσας διπλωματικής. Στη συνέχεια, στο κεφάλαιο 6 αρχικά αναλύονται τα βασικά χαρακτηριστικά και η θεωρία για τους τρεις επιλυτές οι οποίοι χρησιμοποιήθηκαν στην υλοποίηση του προβλήματος της παρούσας διπλωματικής εργασίας. Πιο κάτω στο κεφάλαιο αυτό γίνεται λεπτομερής αναφορά στον τρόπο επίλυσης και υλοποίησης του προβλήματος αυτόνομη και εικονική διαχείριση πόρων στον επιλυτή Minizinc με προγραμματισμό ικανοποίησης περιορισμών και πώς αυτή η υλοποίηση μετατράπηκε για τους επιλυτές Minisat+ και Bsolo για υποστήριξη ψευδοδυαδικών περιορισμών. Εν τέλει, στο κεφάλαιο 7 παρουσιάζονται τα πειράματα τα οποία έχουν πραγματοποιηθεί και τα συμπεράσματα γύρω από τους τρεις επιλυτές υλοποίησης του προβλήματος. Τέλος, μέσα από γραφικές παραστάσεις βασιζόμενες στα πειράματα που πραγματοποιήθηκαν παρουσιάζεται πώς ο χρόνος εκτέλεσης σε κάθε επιλυτή επηρεάζεται από τις παραμέτρους εισόδου του χρήστη στις οποίες θα γίνει αναφορά στο κεφάλαιο αυτό και ποιος από τους τρεις επιλυτές είναι ο γρηγορότερος και έχει τα πιο ικανοποιητικά αποτελέσματα για το πρόβλημα το οποίο έχει υλοποιηθεί.

Κεφάλαιο 2

Υπολογιστικό Νέφος και Εικονοποίηση

2.1 Υπολογιστικό Νέφος

2.1.1 Εισαγωγή στο Υπολογιστικό Νέφος

Υπολογιστικό Νέφος είναι μία δομή με την οποία παρέχεται η δυνατότητα πρόσβασης και χρησιμοποίησης εφαρμογών διαδικτύου χωρίς να τις διαθέτουμε στον υπολογιστή μας ή σε κάποια άλλη συσκευή η οποία είναι διασυνδεδεμένη στο διαδίκτυο. Σε αυτή τη δομή, η εφαρμογή βρίσκεται σε ένα εξυπηρετητή και παρέχεται η δυνατότητα χρησιμοποίησης της χωρίς να χρειάζεται εγκατάσταση στον υπολογιστή. Το υπολογιστικό νέφος αναφέρεται στη χρήση υπολογιστικής ισχύος η οποία βρίσκεται σε ένα «σύννεφο» απομακρυσμένων δικτύων. Αυτή η προσέγγιση, είναι γνωστή σε οποιονδήποτε χρησιμοποιεί διαδικτυακές υπηρεσίες για τη διαχείριση και αποθήκευση δεδομένων, όπως το Hotmail ή το Gmail για ηλεκτρονικό ταχυδρομείο. Συγκεκριμένα για τον υπολογισμό νέφους δίνεται ο εξής ορισμός *«Μεγάλα κέντρα δεδομένων, τα οποία προσφέρουν οικονομίες κλίμακας, φθηνότερη υπολογιστική ισχύ και κυριότερα ευελιξία όσων αφορά την πληρωμή, μόνο για ότι χρησιμοποιεί*

κάποιος». Έτσι, παρέχεται η δυνατότητα στους χρήστες αλλά και στους προγραμματιστές στο να κάνουν περισσότερα με λιγότερα, δηλαδή παρέχεται πρόσβαση σε μεγαλύτερη υπολογιστική ισχύ χωρίς απαραίτητα να χρειάζεται να επενδύσουν μεγάλα χρηματικά ποσά σε εξοπλισμό. Το υπολογιστικό νέφος περιγράφει ένα νέο μοντέλο παροχής υπηρεσιών πληροφορικής, με βάση τα πρωτόκολλα του διαδικτύου και το μοντέλο αυτό συνεπάγεται την τροφοδότηση των δυναμικών, κλιμακωτών και συχνά εικονικών πόρων.

Το υπολογιστικό νέφος τα χαρακτηριστικά του τα υιοθέτησε από τον υπολογισμό χρησιμότητας(utility computing). Επιπλέον όμως παρέχει ένα ελαστικό και δυναμικό περιβάλλον προσφοράς και διάθεσης υπηρεσιών, το οποίο μπορεί να επιτευχθεί με την αυτόματη ανάκαμψη, την αυτό-επιτήρηση, την αυτό-διαχείριση, την αυτόματη επαναδιαμόρφωση και τη δυνατότητα καθορισμού SLAs, τα οποία είναι και τα βασικά του χαρακτηριστικά.[6]

Οι υπηρεσίες που προσφέρει το υπολογιστικό νέφος αναφέρονται ως υπηρεσίες λογισμικού, υπολογισμού, πρόσβασης και αποθήκευσης σε δεδομένα και δεν απαιτούν ο τελικός χρήστης να γνωρίζει τη γεωγραφική θέση και τη διαμόρφωση το συστήματος που παρέχει τις υπηρεσίες.

Οι πάροχοι του υπολογιστικού νέφους διαθέτουν και προσφέρουν εφαρμογές μέσω του διαδικτύου, στις οποίες υπάρχει πρόσβαση μέσω των προγραμμάτων περιήγησης σταθερών υπολογιστών αλλά και από εφαρμογές κινητών, ενώ τα δεδομένα θα είναι αποθηκευμένα σε διακομιστές σε μια απομακρυσμένη τοποθεσία.

2.1.2 Βασικά Χαρακτηριστικά του Υπολογιστικού Νέφους

Πρώτο χαρακτηριστικό του υπολογιστικού νέφους είναι, **κατά Απαίτηση-αυτοεξυπηρέτηση(*On Demand self-service)*.** Συγκεκριμένα, ο καταναλωτής έχει τη δυνατότητα να χρησιμοποιεί την υπολογιστική ισχύ που χρειάζεται όπως συγκεκριμένοι χρόνοι στους εξυπηρετητές ή αποθηκευτικό χώρο σε κάποιο δίκτυο, χωρίς να απαιτείται παρεμβολή ενδιάμεσα μέσω συναλλαγής ή συνεργασίας με κάποιον άνθρωπο ή με τον πάροχο της υπηρεσίας. Ακόμη, ο χρήστης μπορεί ανεξάρτητα και μονομερώς να έχει πρόσβαση σε υπολογιστικές υπηρεσίες όπως συνδεσιμότητα με ένα δίκτυο και αποθήκευση χωρίς να απαιτείται περαιτέρω ανθρώπινη παρέμβαση.[13]

Ένα εξίσου σημαντικό χαρακτηριστικό του υπολογιστικού νέφους είναι η ***Ευρεία πρόσβαση στο δίκτυο***, αφού οι δυνατότητες του είναι διαθέσιμες σε ολόκληρο το δίκτυο και είναι προσβάσεως από τυποποιημένους μηχανισμούς οι οποίοι προωθούν τη χρήση από ετερογενείς πλατφόρμες. Έτσι, όλοι οι πόροι της εφαρμογής είναι διαθέσιμοι από οποιαδήποτε υπολογιστική συσκευή μέσα από όλες τις συνδέσεις στο Διαδίκτυο.

Ένα από τα σημαντικότερα πλεονεκτήματα του υπολογιστικού νέφους είναι η **Ελαστικότητα**. Οι δυνατότητες ενός υπολογιστικού νέφους μπορούν να αποδοθούν αυτόματα και με την ίδια ταχύτητα πρέπει να αποδεσμευτούν. Αν και η ελαστικότητα είναι ένα από τα σημαντικότερα σημεία, αποτελεί ένα από τα δυσκολότερα κομμάτια στην κατασκευή ενός τέτοιου συστήματος. Σημαντικός ρόλος για να λειτουργήσει σωστά η ελαστικότητα είναι η διαχείριση της φυσικής θέσης του υλικού στο δίκτυο κατά την αλλαγή των απαιτήσεων του χρήστη. Εάν δεν υπάρχει σωστή τοποθέτηση, αυτό μπορεί να έχει αρνητικές συνέπειες στις επιδόσεις, την ασφάλεια και την ανθεκτικότητα των συστημάτων.

Ως συνέχεια τον πιο πάνω, ένα ακόμη χαρακτηριστικό του υπολογιστικού νέφους είναι το γεγονός ότι **οι υπηρεσίες είναι μετρήσιμες.** Τα συστήματα νέφους αυτόματα ελέγχουν, διαχειρίζονται και βελτιστοποιούν της διαθέσιμες πηγές τους αξιοποιώντας τη δυνατότητα μέτρησής τους ανάλογα με το είδος της υπηρεσίας.[6,8]

2.1.3 Αρχιτεκτονική του Υπολογιστικού Νέφους

Υπάρχουν τέσσερα επίπεδα (layers) τα οποία ξεχωρίζουν στο υπολογιστικό νέφος και αυτά είναι, το επίπεδο του υλικού, το επίπεδο της υποδομής, το επίπεδο της πλατφόρμας και το επίπεδο της εφαρμογής.

Το **επίπεδο *υλικού***, είναι υπεύθυνο για τη διαχείριση υλικών συσκευών του συστήματος όπως για παράδειγμα εξυπηρετητές. Συγκεκριμένα, αυτό το επίπεδο αφορά το υλικό που υπάρχει στα κέντρα δεδομένων τα οποία αποτελούνται από χιλιάδες εξυπηρετητές που είναι συνδεδεμένοι μεταξύ τους μέσω διακοπτών, δρομολογητών και άλλων υλικών.

Στο **επίπεδο *της υποδομής*** ή αλλιώς εικονικό επίπεδο δημιουργείται μια λίστα από τις διαθέσιμες υπολογιστικές και αποθηκευτικές πηγές πραγματοποιώντας κατάτμηση στο υλικό. Αποτελεί ένα πολύ σημαντικό επίπεδο του υπολογιστικού νέφους αφού πολλές φορές

ορισμένες δυνατότητες όπως η δυναμική απόδοση πόρων πραγματοποιούνται με τεχνολογίες εικονικών μηχανών.

Το ***επίπεδο πλατφόρμας*** είναι πάνω από το επίπεδο υποδομής και αποτελείται από λειτουργικά συστήματα και πλαίσια(Frameworks) λογισμικού. Το επίπεδο αυτό χρησιμεύει στην ελαχιστοποίηση του φόρτου εργασίας από την ανάπτυξη εφαρμογών απευθείας σε μια εικονική μηχανή.

Το ***επίπεδο εφαρμογής***, είναι στην κορυφή σε σχέση με τα άλλα επίπεδα. Στο επίπεδο αυτό βρίσκονται οι εφαρμογές στο νέφος. Η δυνατότητά που έχουν για αυτόματη κλιμάκωση ανάλογα τις απαιτήσεις των καταναλωτών είναι ένα από τα κύρια χαρακτηριστικά που τις ξεχωρίζουν από τις παραδοσιακές εφαρμογές. Με αυτό τον τρόπο πετυχαίνουν καλύτερες επιδόσεις, μεγαλύτερη διαθεσιμότητα και χαμηλό κόστος λειτουργίας.

2.1.4 Μοντέλα υπηρεσιών

Η δυνατότητα που μας δίνεται από το υπολογιστικό νέφος να χρησιμοποιούμε κάποιο λογισμικό μέσα σε αυτό το δίκτυο, ονομάζεται "υπηρεσία". Τα βασικά είδη υπηρεσιών είναι Λογισμικό ως υπηρεσία(Software-as-a-Service-SaaS), Πλατφόρμα ως υπηρεσία(Platform-as-a-Service-PaaS) και υποδομή ως υπηρεσία(Infrastructure-as-a-Service-IaaS). Το καθένα από αυτά τα είδη υπηρεσιών εξυπηρετεί διαφορετικές ανάγκες και προσφέρει διαφορετικές υπηρεσίες.[13]

Το **Λογισμικό ως υπηρεσία(SaaS)**, είναι ίσως το πιο διαδεδομένο μοντέλο ανάπτυξης εφαρμογών στο υπολογιστικό νέφος. Το SaaS, βασίζεται στη λογική της ενοικίασης λογισμικού από έναν πάροχο υπηρεσιών, αντί της αγοράς της άδειας χρήσης καθώς επίσης ότι μια εφαρμογή διαμοιράζεται στους πελάτες από τους εξυπηρετητές του παρόχου της υπηρεσία. Δηλαδή το λογισμικό βρίσκεται σε ένα δίκτυο εξυπηρετητών προκειμένου να διατίθεται ως υπηρεσία από το δίκτυο ή το διαδίκτυο. Σε αυτό τον τύπο υπάρχει μια εφαρμογή η οποία βρίσκεται σε ένα εξυπηρετητή του νέφους και ο χρήστης μπορεί να έχει πρόσβαση σε αυτό μέσω μίας απλή σύνδεσης στο διαδίκτυο. Το λογισμικό αυτό ανήκει σε κάποιον κατασκευαστή και ο χρήστης πληρώνει μόνο ανάλογα με την χρήση που του κάνει και τους πόρους που χρειάζεται και ο κατασκευαστής αναλαμβάνει τα έξοδα συντήρησης του λογισμικού και τη φιλοξενία του σε κάποιον εξυπηρετητή του νέφους.[13]

Το δεύτερο μοντέλο υπηρεσιών είναι το **Πλατφόρμα ως υπηρεσία (PaaS)** το οποίο παρέχει μία πλατφόρμα εφαρμογών για εταιρείες ή ιδιώτες που κατασκευάζουν λογισμικό είτε για ιδία χρήση είτε για τρίτους. Το μοντέλο αυτό παρέχει κατάλληλες υπηρεσίες προκειμένου κάποιος να μπορέσει να αναπτύξει, να διαθέσει και να συντηρήσει εφαρμογές και υπηρεσίες μέσα σε ένα ενιαίο περιβάλλον πλατφόρμας με δυνατότητες αυτό-διαχείρισης και αυτό-συντήρησης της υποδομής, του λειτουργικού συστήματος και της πλατφόρμας εφαρμογών. Δηλαδή, με το PaaS δεν χρειάζεται να ασχολούμαστε με τη συντήρηση του λειτουργικού συστήματος και της πλατφόρμας. Το βασικό του στοιχείο είναι ότι παρέχει την πλατφόρμα την οποία χρησιμοποιεί ένας χρήστης για να δημιουργήσει κάτι όπως για παράδειγμα μια δικτυακή εφαρμογή χωρίς να εγκαταστήσει τίποτα.

Το τρίτο και τελευταίο μοντέλο υπηρεσιών είναι το **υποδομή ως υπηρεσία(IaaS)** το οποίο αναφέρεται στην παροχή υπολογιστικών και δικτυακών υποδομών. Η εταιρεία ή ο ιδιώτης μπορεί να υπενοικιάσει υποδομή(όχι πλατφόρμα) ανάλογα με τις απαιτήσεις χωρίς να χρειαστεί η αγορά εξοπλισμού(υπολογιστικού, δικτυακού) ή σύναψη συμβολαίου παροχής υπηρεσιών φιλοξενίας για συγκεκριμένο χρονικό διάστημα. Το μοντέλο IaaS προσφέρει συνήθως ένα εικονικό υπολογιστικό μηχάνημα που διαχειρίζεται από τον πελάτη. Η διαφορά του με τα υπόλοιπα δύο μοντέλα είναι ότι σε αυτό οι πελάτες έχουν τη δυνατότητα να διαχειρίζονται τις υποδομές που χρειάζονται για τις εφαρμογές τους. Σημαντικό είναι να αναφέρουμε ότι και στα δύο άλλα μοντέλα είναι απαραίτητη η ύπαρξη υποδομών, αλλά η διαχείριση τους δεν είναι ευθύνη ούτε προνόμιο των πελατών τέτοιων υπηρεσιών.

Συγκρίνοντας τα τρία πιο πάνω μοντέλα, στο SaaS μοντέλο του υπολογιστικού νέφους ανήκει το υλικό (όπως για παράδειγμα εξυπηρετητές, αποθηκευτικός χώρος, υποδομές δικτύου) με σκοπό την αξιοποίηση υπολογιστικών, αποθηκευτικών και δικτυακών δυνατοτήτων οι οποίες προσφέρονται ως υπηρεσία στους πελάτες. Στο IaaS, η χρέωση υπολογίζεται ανάλογα τη χρήση των υποδομών και δεν απαιτείται η αγορά τους, με στόχο την παροχή ευέλικτων, πρότυπων και εικονικών περιβαλλόντων τα οποία αποτελούν τα θεμέλια των SaaS και PaaS.

2.1.5 Μοντέλα ανάπτυξης

Τα κυριότερα μοντέλα στο υπολογιστικό νέφος είναι το δημόσιο νέφος(Public Cloud), το ιδιωτικό νέφος(Private Cloud), το κοινοτικό νέφος(Community Cloud) και το υβριδικό νέφος(Hybrid Cloud).

Το Δημόσιο Νέφος(Public Cloud) αποτελεί ένα σύνολο από υπολογιστικούς πόρους οι οποίοι διατίθενται πάνω από το διαδίκτυο και προσφέρονται από έναν πάροχο. Στο μοντέλο εφαρμογής του δημόσιου νέφους οι υποδομές αλλά και οι υπολογιστικές πηγές είναι διαθέσιμες για όλους στο διαδίκτυο. Ένα τέτοιο νέφος αποτελεί ιδιοκτησία αλλά και διαχειρίζεται από τον πάροχό του. Τα βασικά χαρακτηριστικά του νέφους αυτού, είναι ότι η χρέωση της υπηρεσίας αφορά ότι χρησιμοποιηθεί, παρέχει μεγάλη ευελιξία λόγω της άμεσης διάθεσης υπηρεσιών και όλες οι υπηρεσίες προσφέρονται με βελτιωμένη και συνεχή διαθεσιμότητα, ασφάλεια και διαχειρισιμότητα. [13]

Το ιδιωτικό Νέφος(Private Cloud) είναι μια υποδομή του υπολογιστικού νέφους η οποία λειτουργεί αποκλειστικά για έναν οργανισμό. Μπορεί να διαχειρίζεται από τον ίδιο τον οργανισμό ή μια τρίτη οντότητα και επίσης να βρίσκεται είτε εντός του οργανισμού είτε εκτός αυτού. Αυτό το είδος τεχνολογίας νέφους εφαρμόζεται μέσα σε οργανισμούς, εταιρίες όπου δημιουργείται ένα νέφος δίκτυο το οποίο όμως βρίσκεται στα όρια του οργανισμού αυτού. Το δίκτυο αυτό δημιουργείται κατά παραγγελία με βάση τις ανάγκες του οργανισμού. Σημαντικό χαρακτηριστικό του είναι το πολύ υψηλό κόστος απόκτησης και λειτουργίας του. Ως ένα από τα μεγαλύτερα πλεονεκτήματα του ιδιωτικού νέφους είναι τα μειωμένα κόστη των εταιρειών. Επιπρόσθετα υπάρχει ευελιξία στην προσφορά εφαρμογών, ενώ ένα ακόμη χαρακτηριστικό του γνώρισμα είναι η φορητότητα καθώς οι εργαζόμενοι έχουν τη δυνατότητα πρόσβασης στις παρεχόμενες εφαρμογές από ένα μεγάλο εύρος υποστηριζόμενων συσκευών.

Στο μοντέλο του **κοινοτικού νέφους(Community Cloud),** η υποδομή του νέφους μοιράζεται μεταξύ ορισμένων οργανισμών και κοινοτήτων με κοινές ανησυχίες. Μπορεί να διαχειρίζονται από τους ίδιους τους οργανισμούς ή από τρίτες οντότητες που μπορεί να βρίσκονται εντός ή εκτός των εγκαταστάσεων των οργανισμών.[13]

Το **υβριδικό νέφος(Hybrid Cloud)**, είναι ένα μοντέλο εφαρμογής του υπολογιστικού νέφους το οποίο αποτελεί ένα συνδυασμό του δημόσιου και του ιδιωτικού νέφους είτε μέσω συγκεκριμένων προτύπων είτε μέσα από ιδιόκτητη τεχνολογία η οποία επιτρέπει φορητότητα δεδομένων και εφαρμογών. Το πιο σημαντικό σε αυτό το μοντέλο είναι η ενοποίηση δεδομένων από διαφορετικές τοποθεσίες όπως τα ιδιόκτητα κέντρα δεδομένων μιας επιχείρησης μαζί με τα δεδομένα της σε κάποιο ιδιωτικό ή δημόσιο νέφος. Τα

πλεονεκτήματα του υβριδικού νέφους αφορούν τόσο την εξοικονόμηση κόστους όσο και την ευελιξία που δίνεται στις επιχειρήσεις. Σκοπός αυτού του μοντέλου είναι η επίτευξη καλύτερου ελέγχου και ασφάλειας από ένα δημόσιου νέφους σε συνδυασμό με μεγαλύτερη ελαστικότητα από αυτή που προσφέρει ένα ιδιωτικό νέφος. Ουσιαστικά, η τεχνολογία υβριδικού νέφους προσπαθεί να γεφυρώσει τους δύο διαφορετικούς κόσμους του ιδιωτικού και του δημόσιου νέφους.[8]

2.2 Εικονοποίηση

2.2.1 Ορισμός Εικονοποίησης

Η εικονοποίηση είναι ίσως η πρόσφατη τεχνολογική πρόοδος που προσθέτει ένα υψηλότερο επίπεδο στα συστήματα αφού όπως θα δούμε και πιο κάτω έχουν δοθεί πάρα πολλοί ορισμοί για αυτήν. Δίνει την ευκαιρία επίτευξης υψηλότερης παραγωγικότητα όσον αφορά τους υπολογιστές σε μια εργασία. Ένας ορισμός ο οποίος έχει δοθεί είναι, ότι η εικονοποίηση αναφέρεται στην αντικατάσταση μεγάλου αριθμού των εξυπηρετητών που χρησιμοποιείται για τις απαιτήσεις των κέντρων δεδομένων, δεδομένου ότι για κάθε διαφορετική εφαρμογή χρειάζεται και ένας εξυπηρετητής. Μια από τις σημαντικότερες δυνατότητες που προσφέρει είναι να τρέχει πολλαπλά λειτουργικά συστήματα σε ένα μόνο μηχάνημα με αποτέλεσμα τα κέντρα δεδομένων να μπορούν να έχουν πιο λίγους εξυπηρετητές και να ικανοποιούν τις ανάγκες τους.

Με τα σύγχρονα δεδομένα, η σύγχρονη εικονοποίηση επιτρέπει σε πολλαπλές εικονικές μηχανές με τα αντίστοιχα λειτουργικά συστήματα να τρέχουν ταυτόχρονα σε έναν υπολογιστή. Κάθε λειτουργικό σύστημα μοιράζεται τους διαθέσιμους πόρους στο φυσικό υλικό.

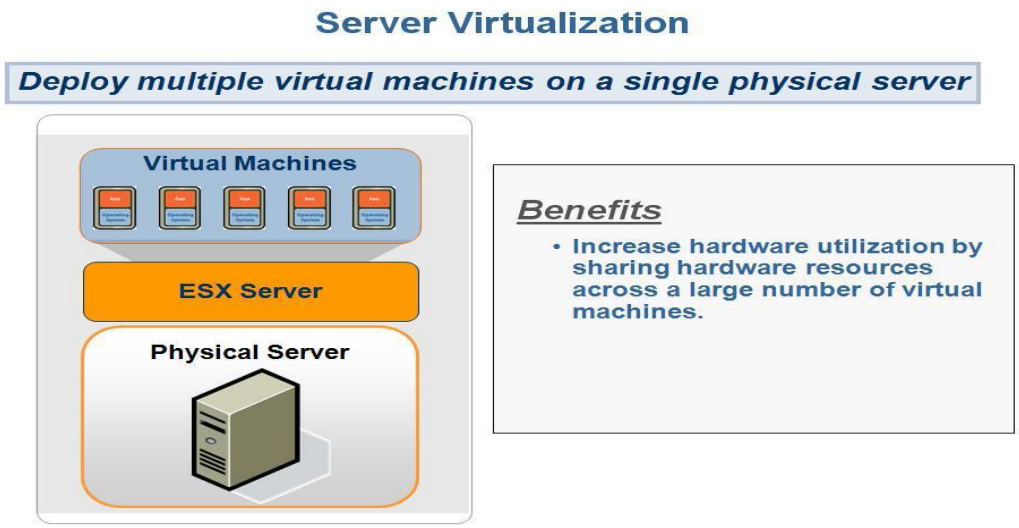
2.2.2 Τύποι Εικονοποίησης

Οι πιο διαδεδομένοι τύποι εικονοποίησης που εφαρμόζονται στα κέντρα δεδομένων είναι η εικονοποίηση διακομιστή(*Server virtualization)* και η εικονοποίηση αποθήκευσης(*Storage Virtualization).*

Η **εικονοποίηση διακομιστή (*Server virtualization)*** είναι μια μεθοδολογία η οποία επιτυγχάνει διαμερισμό των φυσικών πόρων ενός υπολογιστή σε πολλαπλά περιβάλλοντα

εκτέλεσης, εφαρμόζοντας μία ή περισσότερες τεχνολογίες όπως διαμερισμό σε επίπεδο υλικού ή σε επίπεδο λογισμικού και διαμερισμό σε επίπεδο χρόνου. Συγκεκριμένα, είναι η μέθοδος εκτέλεσης πολλών ανεξάρτητων ιδεατών λειτουργικών συστημάτων στον ίδιο φυσικό υπολογιστή. Στην η εικονοποίηση διακομιστή(*Server virtualization)* ένας φυσικός διακομιστής μετατρέπετε σε έναν εικονικό. Ο όρος φυσικός διακομιστής αναφέρεται στο υλικό που πραγματοποιεί την υπολογιστική επεξεργασία που επιβάλλεται από το λογισμικό, όπως το λειτουργικό σύστημα και οι εφαρμογές. Ένας εικονικός διακομιστής δεν μπορεί να λειτουργήσει χωρίς έναν φυσικό. Με την εικονοποίηση διακομιστή είναι δυνατόν πολλοί φυσικοί διακομιστές να μετατραπούν σε εικονικούς και να τοποθετηθούν σε ένα φυσικό διακομιστή ο οποίος ονομάζεται οικοδεσπότης(host). Πολλαπλές εικονικέ μηχανές έχουν την δυνατότητα να μοιράζονται τους φυσικούς πόρους χωρίς να επηρεάζουν το ένα το άλλο έτσι ώστε να μπορούμε με ασφάλεια να τρέξουμε πολλαπλά λειτουργικά συστήματα και εφαρμογές παράλληλα σε έναν υπολογιστή, μοιράζοντάς τον ουσιαστικά σε πολλούς ιδεατούς υπολογιστές(εικονικές μηχανές).

Στην εικόνα 2.2.2, απεικονίζεται η παρουσία πολλαπλών ιδεατών συστημάτων(εικονικές μηχανές) μέσα σε ένα φυσικό σύστημα(φυσικό διακομιστή).



Εικόνα 2.2.2: Απεικόνιση του Server Virtualization

Όσων αφορά την **εικονοποίηση αποθήκευσης(Storage Virtualization)**, παίρνει όλους τους φυσικούς χώρους αποθήκευσης που ανήκουν σε διάφορες συσκευές αποθήκευσης μέσα σε ένα δίκτυο και παρέχει την ψευδαίσθηση ότι είναι μια ενιαία συσκευή αποθήκευσης. Αυτή η συσκευή αποθήκευσης ελέγχεται από μια κεντρική κονσόλα. Με τον όρο εικονοποίηση αποθήκευσης γίνεται αντιληπτή η παρουσίαση φυσικών πολλαπλών αποθηκευτικών

συσκευών. Επίσης αυτός ο τύπος εικονοποίησης παρέχει τα μέσα για τη δημιουργία υψηλού επιπέδου υπηρεσιών αποθήκευσης που κρύβουν την πολυπλοκότητα όλων των εμπλεκόμενων μερών και επιτρέπουν την αυτοματοποίηση της αποθήκευσης δεδομένων. Ο απώτερος στόχος της εικονοποίησης της αποθήκευσης είναι η απλοποίηση της διαχείρισης. Αυτό μπορεί να επιτευχθεί με μια πολύ επίπεδη προσέγγιση, ενώνοντας πολλαπλά τεχνολογικά επίπεδα στο πλαίσιο της λογικής αφαίρεσης.

2.2.3 Ορισμός Εικονικής Μηχανής

Μια εικονική μηχανή είναι ένα απομονωμένο τμήμα λογισμικού που μπορεί να τρέξει τα λειτουργικά συστήματα σαν ένας φυσικός υπολογιστής. Μια εικονική μηχανή συμπεριφέρεται ακριβώς όπως ένας φυσικός υπολογιστής αλλά αντί για ηλεκτρονικά στοιχεία, αποτελείται από ένα σύνολο αρχείων λογισμικού. Μια εικονική μηχανή ονομάζεται «φιλοξενούμενη- guest» ενώ το περιβάλλον μέσα στο οποίο εκτελείται λέγεται «οικοδεσπότης- host». Κάθε εικονική μηχανή αντιπροσωπεύει ένα ολοκληρωμένο σύστημα με επεξεργαστές, μνήμη και αποθηκευτικό χώρο.

Τα τέσσερα σημαντικότερα χαρακτηριστικά των εικονικών μηχανών τα οποία παρέχουν στον χρήστη είναι η απομόνωση, η ενθυλάκωση, η συμβατότητα και η ανεξαρτησία από το υλικό. Με τον όρο **Απομόνωση (isolation)** εννοούμε ότι οι εικονικές μηχανές είναι απομονωμένες η μια από την άλλη σαν ήταν φυσικά διαχωρισμένες. Δεύτερο χαρακτηριστικό των εικονικών μηχανών είναι η **Ενθυλάκωση(encapsulation).** Η ενθυλάκωση παρέχεται στον χρήστη αφού οι εικονικές μηχανές ενθυλακώνουν ένα ολόκληρο υπολογιστικό περιβάλλον. Ένα άλλο εξίσου σημαντικό χαρακτηριστικό τους είναι η **Συμβατότητα(compatibility)** υπό την έννοια ότι οι εικονικές μηχανές είναι συμβατές με όλους τους προτυποποιημένους x86 υπολογιστές. Τελευταίο χαρακτηριστικό τους είναι η **Ανεξαρτησία από το Υλικό(independence)** αφού οι εικονικές μηχανές τρέχουν ανεξάρτητα από το υποκείμενο υλικό.

2.3 Σχέση Υπολογιστικού Νέφους και Εικονοποίησης

Με βάση την πιο πάνω ανάλυση έχει πλέον γίνει αντιληπτό το γεγονός ότι η τεχνολογία της εικονοποίησης παρέχει τη δυνατότητα να ‘‘τρέξουμε’’ εικονικές μηχανές πάνω από έναν ένα στρώμα λογισμικού που εκτελείται απευθείας στο υλικό του υπολογιστή. Κάθε εικονική μηχανή έχει το δικό της λειτουργικό σύστημα, βιβλιοθήκες αλλά και εφαρμογές. Ο hypervisor παρέχει μια ομοιόμορφη αφαίρεση(abstraction) του υποκείμενου υλικού.

Πολλαπλές εικονικές μηχανές μπορούν να εκτελούνται παράλληλα σε έναν hypervisor. Η αποσύζευξη της εικονικής μηχανής με το συγκεκριμένο υλικό είναι αυτό που του επιτρέπει στην ίδια εικονική μηχανή να εκτελεστεί και σε διαφορετικό φυσικό μηχάνημα. Αυτό επιτρέπει στον πάροχο των υπηρεσιών νέφους την απαραίτητη ευελιξία να μετακινεί, να βρίσκει και να διανέμει τους υπολογιστικούς πόρους που του έχουν ζητηθεί όπου υπάρχουν αυτοί διαθέσιμοι στα φυσικά μηχανήματα.

Ωστόσο, εικονοποίηση είναι η διαδικασία της προσομοίωσης εικονικών εκδόσεων των πόρων υποδομής όπως υπολογιστικά περιβάλλοντα, λειτουργικά συστήματα, συσκευές αποθήκευσης σε αντίθεση με τη δημιουργία πραγματικών ή φυσικών εκδόσεων αυτών των ίδιων πόρων. Από την άλλη μεριά, το υπολογιστικό νέφος είναι η παροχή κοινών υπολογιστικών πόρων λογισμικού ή των δεδομένων ως υπηρεσία μέσω του Διαδικτύου, σε αντίθεση με την εικονοποίηση το οποίο αποτελεί μέρος της φυσικής υποδομής. Έτσι η εικονοποίηση είναι η βασική τεχνολογία η οποία τροφοδοτεί το υπολογιστικό νέφος.[14]

Κεφάλαιο 3

Προβλήματα Ικανοποίησης Περιορισμών

3.1 Εισαγωγή

Το κεφάλαιο αυτό ασχολείται με τα προβλήματα ικανοποίησης περιορισμών, τις τεχνικές και του αλγορίθμους που έχουν αναπτυχθεί για τέτοιου είδους προβλήματα. Συγκεκριμένα, αρχικά θα παρουσιαστεί ο ορισμός των προβλημάτων αυτών και στην συνέχεια κάποια παραδείγματα. Ακολούθως, θα γίνει αναφορά στην θεωρία της προτασιακής ικανοποιησιμότητας και εν τέλει θα γίνει αναφορά στη θεωρία γύρω από τους ψευδοδυαδικούς περιορισμούς.

3.2 Ορισμός Προβλημάτων Ικανοποίησης Περιορισμών

Στην κοινότητα της τεχνητής νοημοσύνης, η ικανοποίηση των προβλημάτων με περιορισμούς για πεπερασμένα και μη πεπερασμένα πεδία διατυπώθηκε με τον όρο Προβλήματα ικανοποίησης περιορισμών(CSP). Τυπικά, ένα *πρόβλημα ικανοποίησης περιορισμών*(constraint satisfaction problem CSP) ορίζεται με βάση τα τρία πιο κάτω σημεία:

a. Ένα σύνολο *μεταβλητών X = {X1,X2,. . . . .,Xn}*
b. Για κάθε μεταβλητή *Xi* ένα μη κενό *πεδίο* (domain) *Di* των δυνατών *τιμών* της
c. Ένα σύνολο *περιορισμών C = {C1,C2. . . . .,Cm}*. Κάθε περιορισμός δρα σε ένα σύνολο και καθορίζει τους επιτρεπτούς συνδυασμούς τιμών που μπορούν να πάρουν οι μεταβλητές που ανήκουν στο *σύνολο αυτό*.

Μία κατάσταση (state) του προβλήματος, ορίζεται ως μια *ανάθεση* τιμών σε κάποιες από τις μεταβλητές *{Xi = vi;Xj = vj,....}*. Στην περίπτωση όπου μία ανάθεση δεν παραβιάζει κανέναν από τους περιορισμούς τότε ονομάζεται *συνεπής*(consistent). Μία ανάθεση που περιλαμβάνει όλες τις μεταβλητές ονομάζεται *πλήρης*. *Λύση* του προβλήματος ικανοποίησης περιορισμών είναι μία πλήρης, συνεπής ανάθεση. Συνεπώς, μία λύση του προβλήματος αποτελείται από την πλήρη ανάθεση τιμών στις μεταβλητές όταν ικανοποιούνται όλοι οι περιορισμοί.

3.3 Παραδείγματα Προβλημάτων Ικανοποίησης Περιορισμών

Τα πιο διαδεδομένα παραδείγματα για τα προβλήματα ικανοποίησης περιορισμών είναι το πρόβλημα των N βασιλισσών και το πρόβλημα χρωματισμού χάρτη. Τα δύο αυτά παραδείγματα περιγράφονται πιο κάτω.

Παράδειγμα 1: *Το πρόβλημα χρωματισμού χάρτη*

Το πρόβλημα του χρωματισμού χάρτη(map coloring problem) είναι ένα από τα διαδεδομένα προβλήματα ικανοποίησης περιορισμών. Το πρόβλημα αυτό περιλαμβάνει τον χρωματισμό διαφορετικών περιοχών ενός καθορισμένου χάρτη με περιορισμένο αριθμό χρωμάτων και υπόκειται στον περιορισμό ότι δύο γειτονικές περιοχές δεν επιτρέπεται να έχουν το ίδιο χρώμα. Στην παρούσα διατύπωση, ας θεωρήσουμε τον χάρτη της Αυστραλίας του πιο κάτω σχήματος(Εικόνα 3.3) και ότι έχουμε τα χρώματα κίτρινο, κόκκινο και μπλε.

.

Εικόνα 3.3 : Το πρόβλημα χρωματισμού χάρτη

Για κάθε μια από τις εφτά περιοχές ορίζουμε μία μεταβλητή. Στις μεταβλητές δίνουμε τα εξής ονόματα: *WA, NT, SA, Q, NSW, V, T* και σε κάθε μια από αυτές τις μεταβλητές πρέπει να ανατεθεί ένα χρώμα. Κάθε μεταβλητή έχει ως πεδίο τιμών το σύνολο *{ κίτρινο, κόκκινο, μπλε}*. Ο περιορισμός ο οποίος αναγράφεται πιο κάτω αποτελεί εγγύηση για το γεγονός ότι κάθε ζεύγος γειτονικών περιοχών δεν μπορεί να έχει το ίδιο χρώμα:

$$WA \neq NT \wedge WA \neq SA \wedge NT \neq SA \wedge NT \neq Q \wedge SA \neq Q \wedge$$
$$SA \neq NSW \wedge SA \neq V \wedge Q \neq NSW \wedge NSW \neq V$$

Παράδειγμα 2: *Το Πρόβλημα των N Βασιλισσών*

Ένα δεύτερο πρόβλημα ικανοποίησης περιορισμών αποτελεί το πρόβλημα των *N* βασιλισσών (N-queens). Το πρόβλημα των *N* βασιλισσών στοχεύει στην τοποθέτηση *N* βασιλισσών σε μια σκακιέρα μεγέθους *N*N* σε διαφορετικές θέσεις με τέτοιο τρόπο ώστε καμία βασίλισσα να μην απειλεί κάποια άλλη δηλαδή, να μη βρίσκεται σε ίδια γραμμή, στήλη ή διαγώνιο. Γενικά, θα υπάρχουν *N* μεταβλητές, μία για κάθε γραμμή της σκακιέρας και ουσιαστικά για κάθε βασίλισσα: $X = \{X_1, X_2, \ldots, X_n\}$. Το πεδίο τιμών κάθε μεταβλητής είναι όλες οι πιθανές στήλες στις οποίες μπορεί να τοποθετηθεί η βασίλισσα της συγκεκριμένης γραμμής, *για κάθε i* $D_i = \{1, 2,3,....,N\}$. Ο πιο πάνω τρόπος αναπαράστασης του προβλήματος δεν επιτρέπει δυο βασίλισσες να είναι στην ίδια γραμμή, αφού κάθε γραμμή σημαίνει διαφορετική βασίλισσα. Έτσι οι περιορισμοί που πρέπει να προστεθούν είναι να μην τοποθετηθούν βασίλισσες στην ίδια στήλη ή στην ίδια διαγώνιο.

Για τον περιορισμό ο οποίος αφορά την ίδια στήλη,

$$Xi \neq Xj \text{ Για κάθε } i, j = 1.....N \text{ και } i < j$$

Για τον περιορισμό ο οποίος αφορά την ίδια διαγώνιο,

$$Xi + i \neq Xj + j \text{ και } Xi - i \neq Xj - j \text{ Για κάθε } i, j = 1.....N \text{ και } i < j$$

3.4 Επίλυση Προβλημάτων Ικανοποίησης Περιορισμών

Οι βασικές τεχνικές για την επίλυση προβλημάτων ικανοποίησης περιορισμών είναι οι εξής:

- Αναγωγή Προβλημάτων
- Αναζήτηση
- Σύνθεση Λύσεων

Στην περίπτωση της Αναγωγής Προβλημάτων η βασική ιδέα είναι η αναγωγή κάποιου προβλήματος σε κάποιο ευκολότερο. Οι βασικοί τρόποι αναγωγής είναι η αφαίρεση περιττών τιμών από τα πεδία των μεταβλητών και η σύσφιξη των περιορισμών με την αφαίρεση περιττών ετικετών(label). Η αφαίρεση περιττών τιμών από τα πεδία των μεταβλητών περιλαμβάνει αφαίρεση των τιμών οι οποίες δεν εμφανίζονται σε καμία λύση στο πρόβλημα. Μια ετικέτα(label) είναι μια ταυτόχρονη ανάθεση τιμών σε ένα σύνολο μεταβλητών και ένας περιορισμός πάνω σε ένα σύνολο μεταβλητών είναι ένα σύνολο ετικετών για τις μεταβλητές. Έτσι, το νέο πρόβλημα δυνατόν να είναι ευκολότερο αφού οι μεταβλητές έχουν λιγότερες δυνατές τιμές και υπάρχουν λιγότερες ετικέτες οι οποίες πρέπει να εξεταστούν.

Η περίπτωση της Αναζήτησης, αναφέρεται στον αλγόριθμο οπισθοδρόμησης(backtracking). Ο όρος αναζήτηση με χρονολογική οπισθοδρόμηση, χρησιμοποιείται για να περιγράψει έναν αλγόριθμο αναζήτησης κατά βάθος ο οποίος επιλέγει τιμές για μία μεταβλητή κάθε φορά και κάθε φορά που ένας κλάδος της αναζήτησης αποτυγχάνει, τότε επιστρέφει στην προηγούμενη μεταβλητή και επιλέγει διαφορετική τιμή γι' αυτήν. Αποκαλείται χρονολογική οπισθοδρόμηση γιατί σε κάθε αποτυχία επιστρέφει στο πιο πρόσφατο σημείο διακλάδωσης.

3.5 Είδη Περιορισμών

Ανάλογα με το πόσες μεταβλητές περιλαμβάνει ένας περιορισμός χαρακτηρίζεται ως μοναδιαίος(unary), δυαδικός(binary) ή ανώτερης τάξης(higherorder).

Μοναδιαίος Περιορισμός(Unary Constraint) είναι ο απλούστερος τύπος περιορισμού που υπάρχει ο οποίος περιορίζει την τιμή μίας μοναδικής μεταβλητής. Για παράδειγμα, στο παράδειγμα του χρωματισμού χάρτη μπορεί η μεταβλητή *WA* να μην επιτρέπεται να έχει το χρώμα μπλε(wa≠1), ο αριθμός 1 αντιστοιχεί στο χρώμα μπλε. Κάθε μοναδιαίος περιορισμός μπορεί εξαλειφθεί απλά ελέγχοντας το πεδίο τιμών την αντίστοιχης μεταβλητής και αφαιρώντας όποια τιμή παραβιάζει τον περιορισμό.

Δυαδικός Περιορισμός(Binary Constraint) συσχετίζει τις τιμές δύο μεταβλητών για παράδειγμα, ο περιορισμός *WA ≠ NT στο πρόβλημα του χρωματισμού χάρτη* αποτελεί ένα δυαδικό περιορισμό στον οποίο οι μεταβλητές WA και *NT* δεν μπορούν να έχουν τις ίδιες τιμές, το ίδιο χρώμα αφού είναι γειτονικές. Οι αναπαράσταση δυαδικών περιορισμών μπορεί να γίνει και με γράφο.

Υψηλού βαθμού Περιορισμός(High-order Constraint) συσχετίζουν τις τιμές τριών και περισσοτέρων μεταβλητών.

3.6 Προτασιακή Ικανοποιησιμότητα

Το πρόβλημα της προτασιακής ικανοποιησιμότητας ορίζεται σε ένα σύνολο από δυαδικές μεταβλητές και ένα σύνολο από περιορισμούς. Οι περιορισμοί αυτοί είναι διατυπωμένοι σε κανονική συζευκτική μορφή(Conjunctive Normal Form). Στόχος είναι η ανάθεσης τιμών έτσι ώστε να ικανοποιούνται όλοι οι περιορισμοί ή η απόδειξη ότι είναι αδύνατον η εύρεση μιας τέτοιας ανάθεσης. Τα προβλήματα της προτασιακής ικανοποιησιμότητας επιλύονται με την χρήση εργαλείων(SAT επιλυτών) τα οποία επιλύουν πρακτικά προβλήματα και τα τελευταία χρόνια έχουν διαδοθεί πάρα πολύ.[2,11]

Η γλώσσα αναπαράστασης των προβλημάτων προτασιακής ικανοποιησιμότητας είναι η προτασιακή λογική. Τα σύμβολα της προτασιακής λογικής είναι οι σταθερές true(αληθές) και false(ψευδές), ένα αριθμήσιμο απειροσύνολο προτασιακών συμβόλων και οι λογικοί σύνδεσμοι ∧ (and), ∨(or) ,¬ (not) , →(implication) , ↔ (equivalence).

Ένας λογικός τύπος είναι σε Κανονική Συζευκτική Μορφή(CNF) αν αποτελείται από φράσεις συνδεδεμένες με συζεύξεις (∧) όπως για παράδειγμα: $(x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_3 \vee x_5 \vee x_6) \wedge (x_3 \vee x_6)$. Συγκεκριμένα μια cnf πρόταση περιέχει δυαδικές μεταβλητές $x_1,..,x_n$, και η κάθε μια από αυτές είναι μια σύζευξη (AND) m όρων (clauses) $y_1,\ldots,y_m$ και ο καθένας από τους όρους αυτούς αποτελείται από τη διάζευξη (OR) *k* λεκτικών (literals). Όπως αναφέρθηκε και σε προηγούμενο υπο-κεφάλαιο, λεκτικό (literal) είναι η εμφάνιση (x_i) ή το συμπλήρωμα ($\neg x_i$) μιας δυαδικής μεταβλητής. Οι μεταβλητές έχουν ως πεδίο τιμών τις τιμές {0,1}. Το κάθε λεκτικό επίσης έχει ως πεδίο τιμών τις τιμές {0,1} και αποτιμάται ως ψευδής ή αληθής αντίστοιχά. Ένα όρος (clause) επίσης αποτιμάται ως αληθής και λέμε ότι είναι ικανοποιήσιμος εάν έστω και ένα λεκτικό του πάρει την τιμή 1, δηλαδή αληθεύει (με βάση τον πίνακα αληθείας της διάζευξης). Στην περίπτωση όπου όλα τα λεκτικά τα οποία αποτελούν έναν όρο (clause) είναι ψευδή τότε λέμε ότι ο όρος (clause) αυτός δεν είναι ικανοποιήσιμος και αποτιμάται ως ψευδής. Σε περίπτωση που τα λεκτικά (literal) είναι όλα ψευδή εκτός από ένα τότε η διαζευκτική πρόταση ονομάζεται μοναδικό λεκτικό (unit literal). Λέμε ότι μια πρόταση είναι ικανοποιήσιμη εάν όλες οι διαζευκτικές προτάσεις της είναι ικανοποιήσιμες. Μια πρόταση είναι μη ικανοποιήσιμη όταν έστω και μια διαζευκτική πρόταση της δεν ικανοποιείται(με βάση τον πίνακα αλήθειας της σύζευξης).[2,11]

Τα περισσότερα εργαλεία προτασιακής ικανοποιησιμότητας(SAT επιλυτές) στηρίζονται στον αλγόριθμό Davis – Putnam –Logemann – Loveland (DPLL). Στην αρχή της εκτέλεσης του αλγορίθμου οι μεταβλητές δεν έχουν κάποια τιμή. Έπειτα γίνεται ανάθεση σε μια μεταβλητή και ελέγχεται η συνέπεια με τις υπόλοιπες μεταβλητές. Αν ο αλγόριθμος δεν εντοπίσει κάποια σύγκρουση συνεχίζει με την δημιουργία μιας νέας ανάθεσης τιμών σε μια άλλη μεταβλητή στην οποία δεν έγινε μέχρι στιγμής ανάθεση τιμής. Στην περίπτωση όπου εντοπιστεί σύγκρουση, τότε ο αλγόριθμος επιστρέφει προς τα πίσω (οπισθοδρόμηση-backtracking) και αφαιρεί από την μεταβλητή την δοθείσα τιμή και συνεχίζει με την εύρεση καινούργιας τιμής.

3.7 Ψευδοδυαδικοί Περιορισμοί

3.7.1 Γενικά

Οι ψευδοδυαδικοί περιορισμοί και οι ψευδοδυαδικές συναρτήσεις τον τελευταίο καιρό έχουν μεγάλη εφαρμογή σε ποικίλα προβλήματα για τον λόγο ότι έχουν μεγαλύτερη ευελιξία στην έκφραση. Η ανάθεση στις τιμές των μεταβλητών γίνεται από το δυαδικό σύνολο {0,1} και η

αποτίμηση μιας πρότασης με ψειδοδυαδικούς περιορισμούς επεκτείνεται σε όλο το σύνολο των ακέραιων αριθμών.[2]

Οι ψευδοδυαδικές συναρτήσεις είναι συναρτήσεις στις οποίες οι δυαδικές τιμές αντιστοιχούν σε πραγματικούς αριθμούς. Πολλά προβλήματα σε διάφορου τομείς μπορούν να εκφραστούν ως προβλήματα βελτιστοποίησης της τιμής μιας ψευδοδυαδικής συνάρτησης. Οι Ψευδοδυαδικοί περιορισμοί είναι πιο εκφραστικοί αφού μπορούν να δείξουν με ένα πιο φυσικό τρόπο τους περιορισμούς σε σύγκριση με τους όρους (clauses). Όρος (clause) είναι μια διάζευξη από λεκτικά της μορφής $x \vee \neg y \vee w$.

Η ανάθεση τιμών στις μεταβλητές γίνεται με δύο τιμές, την τιμή 1 ή την τιμή 0, αληθής ή ψευδής αντίστοιχα. Κάθε όρος i σε κάθε πρόταση παίρνει την τιμή x_i ή ┐x_i (x_i-1). Ο θετικός όρος είναι ο x_i κα ισοδυναμεί με αληθές εφόσον το x_i είναι αληθές. Ο αρνητικός όρος είναι ο ┐x_i και ισοδυναμεί με ψευδές εφόσον το x_i είναι ψευδές.[2,3]

Στους ψευδοδυαδικούς περιορισμοί ένας όρος παίρνει τιμή 0 ή 1 και αποτιμάται με ψευδές ή αληθές αντίστοιχα. Αφού έχει γίνει η αποτίμηση πολλαπλασιάζεται με ένα ακέραιο αριθμό, τον συντελεστή του συγκεκριμένου όρου.

3.7.2 Γραμμικοί Ψευδοδυαδικοί περιορισμοί

Ένας γραμμικός Ψευδοδυαδικός περιορισμός (LPB Constraint)έχει την μορφή $a_k*L_k + a_k+*L_{k+1} +, \ldots..,+ a_n*L_n \otimes t$, όπου k,n ανήκουν στους φυσικούς αριθμούς και για κάθε αριθμό στους φυσικούς i τ.ω το i να είναι ανάμεσα του k και του n, L_i είναι μια μεταβλητή(literal - λεκτικό) της μορφής x_i ή ┐x_i (x_i-1) και το x_i έχει πεδίο τιμών {0,1}. Τα a_k και το t είναι ακέραιες μεταβλητές. Το σύμβολο αριστερά από το t αντιστοιχεί στους σχεσιακούς τελεστές (=, <, >, ≤, ≥). Το δεξί μέρος του περιορισμού είναι ο βαθμός του περιορισμού. Οι τιμές 0 ή1 λαμβάνονται από το x_i εάν η δυαδικοί μεταβλητή είναι ψευδής ή αληθής. Ο περιορισμός θεωρείται ότι είναι ικανοποιήσιμος όταν το δεξί και αριστερό μέρος του περιορισμού πάρουν τιμές ακεραίων τέτοιες ώστε να ικανοποιούν τον σχεσιακό τελεστή. [11]

Παραδείγματα γραμμικών Ψευδοδυαδικών περιορισμών είναι τα πιο κάτω:

1. $3x_1 + 5x_2 \geq 8$

   Ο περιορισμός αυτός ικανοποιείται μονό για $x_1=1$ και $x_2=1$. Για οποιονδήποτε άλλο συνδυασμό δεν μπορεί να ικανοποιηθεί.

2. $4x_1 + x_2 \geq 3$

   Ο περιορισμός αυτός ικανοποιείται για $x_1=1$ και $x_2=0$ αλλά δεν ικανοποιείται για τις τιμές $x_1=0$ και $x_2=0$.

3.7.3 Μη Γραμμικοί Ψευδοδυαδικοί περιορισμοί

Ένας μη γραμμικός Ψευδοδυαδικός περιορισμός έχει την μορφή $a_k*(L_k*q_1+..+L_k*q_m) + a_k+*(L_{k+1}*s_1+..+L_{k+1}*s_r) +, \ldots..,+ a_n*(L_n*z_1+..+L_n*z_b) \otimes t$, όπου k, n ανήκουν στους φυσικούς αριθμούς και για κάθε αριθμό στους φυσικούς i τ.ω το i να είναι ανάμεσα του k και του n, L_i είναι μια μεταβλητή (literal - λεκτικό) της μορφής x_i ή ┐x_i (x_i-1) και το x_i έχει πεδίο τιμών {0,1}. Τα a_k και το t είναι ακέραιες μεταβλητές. Το σύμβολο αριστερά από το t αντιστοιχεί στους σχεσιακούς τελεστές(=, <, >, ≤, ≥). Σε ένα τέτοιο ψευδοδυαδικό περιορισμό το γινόμενο του κάθε αθροίσματος περιλαμβάνει ένα σύνολο από όρους, όπου ο κάθε όρος του γινομένου αυτού είναι δυνατόν να πάρει μόνο τις δυαδικές τιμές {0,1}. Το αποτέλεσμα του γινομένου αυτό είναι ισοδύναμο με την λογική πράξη της σύζευξης. Αυτού του είδους περιορισμός ικανοποιείται όταν στο δεξί και στο αριστερό μέρος γίνει ανάθεση ακέραιων τιμών τέτοια ώστε να ικανοποιείται ο σχεσιακός τελεστής (=, <, >, ≤, ≥).

Παραδείγματα μη γραμμικών ψευδοδυαδικών περιορισμών είναι τα πιο κάτω:

1. $4x_1+2x_1.x_2 \geq 3$

   Ο περιορισμός αυτός ικανοποιείται για $x_1=1$ και $x_2=0$ αλλά δεν ικανοποιείται για τις τιμές $x_1=0$ και $x_2=1$ ή $x_2=0$.

2. $x_1.x_2.x_3 + ┐x_4 \geq 1$

   Ο περιορισμός αυτός ικανοποιείται για $x_1=x_2=x_3=x_4=1$ αλλά δεν ικανοποιείται για τις τιμές $x_1=x_2=x_3=0$ και $x_4=1$.

3.7.4 Κανονικοποίηση Ψευδοδυαδικών περιορισμών

Το γεγονός ότι υπάρχουν πολλές διαφορετικές συντάξεις οι οποίες είναι όλες ισοδύναμες, καθιστά τους ψευδοδυαδικούς περιορισμούς να βρίσκονται σε υψηλό επίπεδο μη-κανονικής μορφής. Για αυτό τον λόγο θα πρέπει να εφαρμοστούν κάποιες τεχνικές έτσι ώστε να υπάρχει μια κανονική μορφή των ψευδοδυαδικών περιορισμών. Οι τεχνικές αυτές αναφέρονται στην κανονικοποίηση των ψευδοδυαδικών περιορισμών, με σκοπό να υπάρχει ένας κοινός συμβολισμός ανάμεσα σε όλους τους περιορισμούς. Η κανονικοποίηση είναι βοηθητική στο να παρθεί απόφαση κατά πόσο μπορεί να ικανοποιηθεί ή όχι μια πρόταση. Στην περίπτωση κατά την οποία μια πρόταση είναι ικανοποιήσιμη και βρίσκεται στην κανονικοποιημένη της μορφή τότε, βοηθά τον επιλυτή για να ψάξει για ανάθεση στους όρους της πρότασης πιο γρήγορα.[9]

Η μορφή κανονικοποίησης ενός περιορισμού είναι $\sum_{j=1}^{n} a_j * l_j \geq b$, τα a_j, b είναι μη αρνητικοί φυσικοί αριθμοί και το l_j είναι της μορφής x_j ή ¬ x_j. Πιο κάτω παρουσιάζονται οι κανόνες με τους οποίους ένας ψειδοδυαδικούς περιορισμός μπορεί να κανονικοποιηθεί.

1. Οι περιορισμοί που εκφράζονται από την ανισότητα ≤ - μετατρέπονται σε ≥ - περιορισμούς αναιρώντας σταθερές. Αν ο περιορισμός έχει το σύμβολο ≤ τότε πολλαπλασιάζουμε και τα δυο μέρη με -1 και αντικαθιστούμε τον τελεστή με ≥.
2. Αν ο τελεστής είναι >, τότε προσθέτουμε 1 από το δεξιό μέλος και αλλάζουμε τον τελεστή σε ≥.
3. Αν ο τελεστής είναι = τότε αντικαθιστούμε τον περιορισμό, με δύο ίδιους περιορισμούς, ο ένας με τον τελεστή ≥ και ο άλλος με τον τελεστή ≤.
4. Αρνητικοί συντελεστές αφαιρούνται αλλάζοντας το q σε ¬q και ενημερώνοντας την δεξιά πλευρά.
5. Οι συντελεστές ταξινομούνται κατά αύξουσα σειρά: $C_i \leq C_j$ αν i < j.
6. Συντελεστές μεγαλύτεροι από την δεξιά πλευρά αντικαθιστούνται από την δεξιά πλευρά.
7. Οι συχνές εμφανίσεις της ίδιας μεταβλητής συγχωνεύονται σε ένα όρο C_i x ή C_i ¬x.
8. Κοινότυποι περιορισμοί όπως “x + y ≥ 0” διαγράφονται.
9. Κοινότυποι ανικανοποίητοι περιορισμοί (“x + y ≥ 3”) απορρίπτονται και δίδεται απάντηση ότι το πρόβλημα δεν ικανοποιείται.
10. Οι συντελεστές της αριστερής πλευράς χωρίζονται με βάση το μέγιστο κοινό διαιρέτη τους.

3.8 Προβλήματα αποφάσεων (decision problem) και προβλήματα βελτιστοποίησης(optimization problem)

Υπάρχουν δύο είδη προβλημάτων τα Προβλήματα αποφάσεων και τα προβλήματα βελτιστοποίησης. Όσον αφορά τα προβλήματα αποφάσεων η απάντηση σε αυτά είναι μόνο μια. Συγκεκριμένα γίνεται ο συνδυασμός Ψευδοδυαδικών περιορισμών και αναγνώριση κατά πόσο υπάρχει μια ανάθεση τιμών τέτοια ώστε το πρόβλημα να θεωρείται ικανοποιήσιμο δηλαδή, να ικανοποιούνται οι περιορισμοί του προβλήματος. Η μοναδική απάντηση που δίδεται ως έξοδος είναι κατά πόσο το πρόβλημα είναι ικανοποιήσιμο ή μη ικανοποιήσιμο, αν μπορούν δηλαδή να ικανοποιηθούν όλοι οι περιορισμοί του προβλήματος ταυτόχρονα ή όχι αν αυτό δεν είναι εφικτό. Όσον αφορά τα προβλήματα βελτιστοποίησης, εξετάζεται εάν ένα πρόβλημα έχει λύση ή όχι. Στη περίπτωση όπου το πρόβλημα μπορεί να επιλυθεί, δηλαδή υπάρχει λύση κατά την οποία ικανοποιούνται όλοι οι περιορισμοί του προβλήματος, τότε γίνεται προσπάθεια εύρεσης τις πιο ικανοποιητικής λύσης - βέλτιστης λύσης. Δηλαδή η απάντηση που δίνεται ως έξοδος περιλαμβάνει την ικανοποίηση του προβλήματος μαζί με την βέλτιστη λύση ή σε αντίθετη περίπτωση όπου προκύπτει μη επίλυσης δίνεται ως έξοδος η μη ικανοποίηση του προβλήματος.[3,10]

Ως εκ τούτου ένα πρόβλημα δυνατόν να έχει αρκετές βέλτιστες λύσεις. Για τον λόγο ότι υπάρχουν αρκετές λύσεις στο πρόβλημα της βελτιστοποίησης πρέπει να εντοπίζεται ποία λύση είναι καλύτερη από κάποια άλλη. Αυτό μπορεί να επιτευχθεί με μια συνάρτηση την αντικειμενική συνάρτηση(objective function) η οποία αντιστοιχεί κάθε λύση με ένα ακέραιο. Η συνάρτηση αυτή μπορεί να ελαχιστοποιήσει ή να μεγιστοποιήσει την τιμή της ανάλογα. Στην περίπτωση ελαχιστοποίησης αναφέρεται ως συνάρτηση κόστους(cost function) και η λύση S είναι η βέλτιστη σε σχέση με την λύση S' εάν ικανοποιείται η σχέση *ObjectiveFunction(S)< ObjectiveFunction(S').* Στην περίπτωση μεγιστοποίησης αναφέρεται ως συνάρτηση ωφέλειας(utility function) και η λύση S είναι η βέλτιστη σε σχέση με την λύση S' εάν ικανοποιείται η σχέση *ObjectiveFunction(S)>ObjectiveFunction(S').*[3]

Τέλος, κάτι το οποίο είναι σημαντικό είναι το γεγονός ότι υπάρχει η δυνατότητα όλα τα προβλήματα μεγιστοποίησης να μετατραπούν σε προβλήματα ελαχιστοποίησης. Συγκεκριμένα όλα τα προβλήματα ορίζονται με την ελαχιστοποίηση της βέλτιστης συνάρτησης εφόσον η μεγιστοποίηση της βέλτιστης συνάρτησης είναι ισοδύναμη με την άρνηση της ελάχιστης συνάρτησης.[10]

Κεφάλαιο 4

Διαφορετικές Προσεγγίσεις Του Προβλήματος - Αλγόριθμοι Τοποθέτησης Εικονικών Μηχανών στο Υπολογιστικό Νέφος

4.1 Εισαγωγή

Η τοποθέτηση εικονικών μηχανών είναι η διαδικασία κατά την οποία οι εικονικές μηχανές τοποθετούνται σε φυσικές μηχανές. Με άλλα λόγια, η τοποθέτηση αυτή είναι η διαδικασία επιλογής του καταλληλότερου «οικοδεσπότη» για την εικονική μηχανή. Η διαδικασία αυτή περιλαμβάνει κατηγοριοποίηση του υλικού των εικονικών μηχανών, των απαιτήσεων σε πόρους και την αναμενόμενη χρήση των πόρων με στόχο την τοποθέτηση τους. Ο στόχος τοποθέτησης μπορεί να είναι είτε μεγιστοποίηση της χρήσης των διαθέσιμων πόρων ή ακόμη μπορεί να περιλαμβάνει εξοικονόμηση ενέργειας με την έννοια ότι μπορεί να

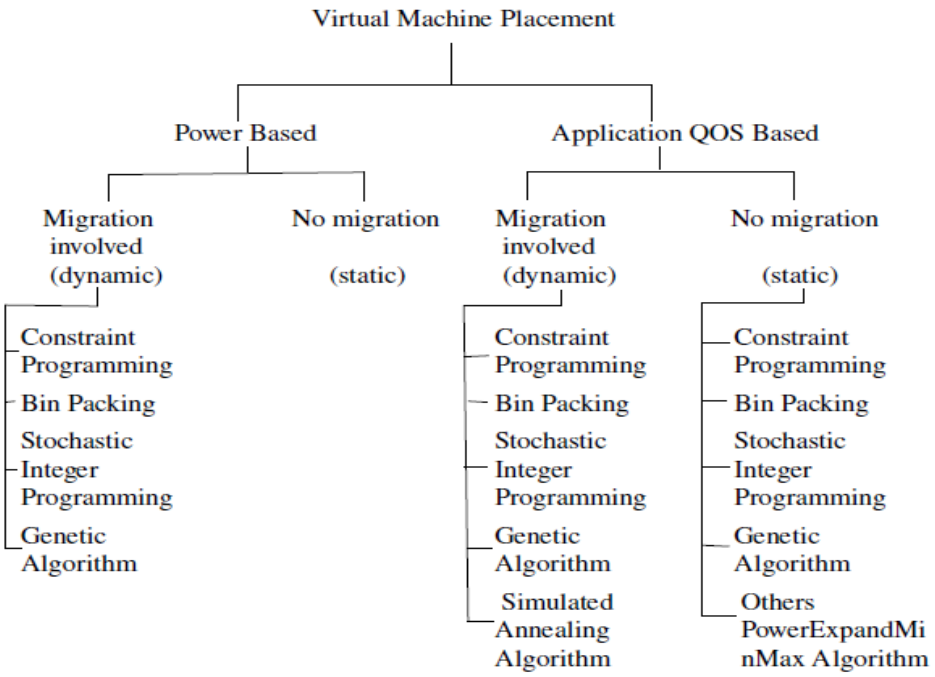
απενεργοποιηθούν μερικοί από τους εξυπηρετητές. Οι αυτόνομοι αλγόριθμοι τοποθέτησης εικονικών μηχανών είναι σχεδιασμένοι έχοντας κατά νου τους παραπάνω στόχους.
Στη συνέχεια του κεφαλαίου αυτού θα γίνει αναφορά σε μερικούς αλγόριθμους οι οποίοι έχουν σχεδιαστεί για την αντιμετώπιση του προβλήματος τοποθέτησης εικονικών μηχανών.

Οι αλγόριθμοι τοποθέτησης μπορεί να είναι σε γενικές γραμμές μια κατάταξη σε δύο κατηγορίες με βάση το στόχο τοποθέτησής τους:

- Βασιζόμενοι στην Ισχύ: Ο κύριος στόχος αυτών των προσεγγίσεων είναι η τοποθέτηση των εικονικών μηχανών σε φυσικές μηχανές με τέτοιο τρόπο, έτσι ώστε μερικοί από τους εξυπηρετητές να αξιοποιήσουν στο μέγιστο την αποδοτικότητα τους και οι υπόλοιποι εξυπηρετητές να είναι είτε σε νάρκη ή να κλείσουν αναλόγως.

- Βασιζόμενοι στο Qos των εφαρμογών: Αυτοί οι αλγόριθμοι διαχειρίζονται τη τοποθέτηση των εικονικών μηχανών σε φυσικές μηχανές, με στόχο τη μεγιστοποίηση της ποιότητας των υπηρεσιών(QoS) που παραδίδονται.[1]

Αυτή η κατάταξη των αλγορίθμων παρουσιάζεται στην πιο κάτω εικόνα.



4.2 Προσεγγίσεις Τοποθέτησης

Υπάρχουν πολλές προσεγγίσεις οι οποίες αφορούν το πρόβλημα τοποθέτησης των εικονικών μηχανών. Μερικές από αυτές εξηγούνται μέσα από τις επόμενες παραγράφους. Στην συνέχεια, γίνεται αναφορά σε μερικούς από τους αλγορίθμους που έχουν χρησιμοποιηθεί για να λυθεί το πρόβλημα της τοποθέτησης εικονικών μηχανών.

Οι προσεγγίσεις επίλυσης το προβλήματος τοποθέτησης στις οποίες θα αναφερθούμε είναι οι εξής:

1. Προγραμματισμός με ικανοποίηση περιορισμών
2. Πρόβλημα Τοποθέτησης σε Κάδους (Bin Packing)
3. Στοχαστικός Ακέραιος Προγραμματισμός
4. Γενετικός Αλγόριθμος

Το κίνητρο πίσω από όλους τους αλγορίθμους οι οποίοι έχουν σχεδιαστεί για τη διάθεση εικονικών μηχανών στις διαθέσιμες φυσικές μηχανές, είναι η μείωση του κόστος που εμπλέκονται. Το κόστος αυτό μπορεί να μειωθεί με τη βέλτιστη χρήση των διαθέσιμων πόρων αλλά και κλείνοντας τους εξυπηρετητές οι οποίοι δεν έχουν φορτίο.

4.2.1 Προγραμματισμός με Περιορισμούς

Όπως θα γίνει αναφορά και σε επόμενο κεφάλαιο ο προγραμματισμός με περιορισμούς είναι ένα παράδειγμα(paradigm) προγραμματισμού. Η λύση θα πρέπει να ικανοποιεί τους περιορισμούς σχετικά με τις σχέσεις μεταξύ των μεταβλητών. Ένα πρόβλημα ικανοποίησης περιορισμών αποτελείται από τρεις βασικές τεχνικές:

- Εύρεση των μεταβλητών.
- Αναγνώριση των πεδίων τιμών των μεταβλητών.
- Εντοπισμός των περιορισμών σχετικά με τις δηλωμένες μεταβλητές.

Το πρόβλημα της τοποθέτησης εικονικών μηχανών μπορεί να σχεδιαστεί ως ένα πρόβλημα προγραμματισμού με περιορισμούς. Το πρόβλημα ορίζεται ως εξής:

Περιορισμός: Αριθμός διαθέσιμων φυσικών μηχανών και μεγιστοποίηση της ικανοποίησης κάθε εφαρμογής(global utility function).

Μοντέλο: Εύρεση διανύσματος κατανομής εικονικών μηχανών.

Στη συνέχεια του υπο-κεφαλαίου αυτού, θα γίνει λεπτομερής αναφορά σε μια προσέγγιση του προβλήματος τοποθέτησης εικονικών μηχανών με την χρήση προγραμματισμού με

περιορισμούς. Συγκεκριμένα το πρόβλημα αυτό αναφέρεται ως «αλγόριθμος τοποθέτησης εικονικών μηχανών στο κέντρο δεδομένων».

Μια άλλη προσέγγιση με προγραμματισμό με περιορισμούς είναι η «Αυτόνομη Εικονική Διαχείριση Πόρων». Αυτή η προσέγγιση θα αναλυθεί επίσης λεπτομερώς στο κεφάλαιο 5 η οποία είναι η προσέγγιση η οποία υλοποιήθηκε για τους σκοπούς της παρούσας διπλωματικής.

4.2.1.1 Αλγόριθμος τοποθέτησης εικονικών μηχανών στο κέντρο Δεδομένων

Ο προγραμματισμός με περιορισμούς για την τοποθέτηση εικονικών μηχανών μπορεί να μειώσει αποτελεσματικά τον αριθμό των φυσικών μηχανών για την επίτευξη του στόχου, της μείωσης της λειτουργικού κόστους και την βελτίωση της χρήσης των πόρων. Σε αυτό το μέρος του κεφαλαίου θα παρουσιαστεί ένας αλγόριθμος τοποθέτησης εικονικών μηχανών με βάση των προγραμματισμό ικανοποίησης περιορισμών, με στόχο την ελαχιστοποίηση του αριθμού των φυσικών μηχανών οι οποίες φιλοξενούν εικονικές μηχανές.

Οι πλατφόρμες του υπολογιστικού νέφους προσφέρουν στους πελάτες αποθήκευση, υπολογιστική δύναμη, ανάπτυξη περιβάλλοντος και υπηρεσίες πληροφορικής σε διαφορετικά στρώματα. Η ανάπτυξη ικανότητας υπολογιστικού νέφους είναι στενά συνδεδεμένη με την υποστήριξη υποκείμενων κέντρων δεδομένων τα οποία παρέχουν ισχυρή και παράλληλη υπολογιστική δύναμη και μαζική αποθήκευση δεδομένων.[15]

Το λειτουργικό κόστος αυξάνεται ραγδαία. Ωστόσο τα κέντρα δεδομένων των πόρων είναι σοβαρά υπο-χρησιμοποιούμενα. Ο κύριος λόγος υπο-χρησιμοποίησης είναι το γεγονός ότι ο διαχειριστής κατανέμει πόρους υλικού για την φιλοξενία εφαρμογών σύμφωνα με τις μέγιστες απαιτήσεις των εφαρμογών. Έτσι είναι μεγάλη πρόκληση να μειωθεί το λειτουργικό κόστος και να βελτιωθεί η χρησιμοποίηση των πόρων ενώ ικανοποιούνται τα SLA των εφαρμογών, δηλαδή η συμφωνία μεταξύ δύο ή περισσότερων μέρων, όπου το ένα είναι ο πελάτης και οι άλλοι είναι πάροχοι υπηρεσιών.

Η τεχνολογία της εικονοποίησης διακομιστή είναι μια αποτελεσματική προσέγγιση για την βελτίωση της αποδοτικότητας των πόρων και της εξοικονόμησης ενέργειας, ενώ παράλληλα παρέχει εγγυήσεις επίδοσης για τις εφαρμογές. Έτσι, χρησιμοποιώντας την τεχνολογία αυτή ο χρόνος εκτέλεσης των εφαρμογών εγκλείεται σε μια εικονική μηχανή και μία ή

περισσότερες εικονικές μηχανές φιλοξενούνται σε μια φυσική μηχανή. Επίσης η τεχνολογία της εικονοποίησης μπορεί να παρέχει απομόνωση επίδοσης μεταξύ εικονικών μηχανών που τοποθετούνται στην ίδια φυσική μηχανή, και να αποφεύγεται η υποβάθμιση επίδοσης που προκαλείται από άπληστες ή κακόβουλες εφαρμογές. Εξάλλου, είναι χρήσιμο να βελτιωθεί η αξιοποίηση των πόρων και η εξοικονόμηση κόστους. Επιπλέον η τεχνολογία εικονοποίησης επιτρέπει την ζωντανή μετανάστευση των εικονικών μηχανών έτσι που να πληρούν τις απαιτήσεις επίδοσης των εφαρμογών.[15]

Ωστόσο, χρειάζεται μια αυτοματοποιημένη και ευφυή προσέγγιση για την αντιμετώπιση και την διαχείριση των εικονικών μηχανών. Η διαχείριση των εικονικών μηχανών αποτελείται από δύο φάσεις, την φάση αρχικού σχεδίου(Initial plan) και την φάση διαχείριση της εκτέλεσης. Στη πρώτη φάση, δεδομένα είναι ένα σύνολο από εικονικές μηχανές και απαιτούμενους πόρους από κάθε εικονική μηχανή καθώς και ένα σύνολο από φυσικές μηχανές οι οποίες φιλοξενούν όλες τις εικονικές μηχανές. Στόχος της φάσης αυτής είναι η ελαχιστοποίηση του αριθμού των φυσικών μηχανών που φιλοξενούν όλες τις εικονικές μηχανές. Στην δεύτερη φάση, πρέπει να ρυθμιστεί η διαμόρφωση των εικονικών μηχανών ή να εκτελεστεί μια ζωντανή μετανάστευση των εικονικών μηχανών σύμφωνα με τα δυναμικά φορτία. Σκοπός της φάσης αυτής είναι να ελαχιστοποιηθεί ο αριθμός των μεταναστεύσεων ενώ πληρούν τα SLA των εφαρμογών. Στη συνέχεια αυτού του κεφαλαίου θα εστιάσουμε στην πρώτη φάση, δηλαδή στο πρόβλημα τοποθέτησης εικονικών μηχανών με στόχο την ελαχιστοποίηση του αριθμού των ενεργών φυσικών μηχανών.[15]

Διατύπωση του προβλήματος Τοποθέτησης εικονικών μηχανών - Βασιζόμενος στον Προγραμματισμό με περιορισμούς

Στο πρόβλημα τοποθέτησης εικονικών μηχανών, οι εικονικές μηχανές εμφανίζονται ως κουτιά όπου διάφορα αιτήματα πόρων από κάθε εικονική μηχανή θεωρούνται ως διαστάσεις του κουτιού με μη αρνητικές τιμές. Οι φυσικές μηχανές θεωρούνται ως κάδοι και τα cpu, μνήμη και εύρος ζώνης(bandwith) θεωρούνται ως ιδιοκτησία των κουτιών. Ο στόχος του προβλήματος τοποθέτησης εικονικών μηχανών είναι να προσδιοριστεί ο ελάχιστος αριθμός των φυσικών μηχανών που απαιτούνται για το σύνολο των εικονικών μηχανών.[15]Το πρόβλημα τοποθέτησης εικονικών μηχανών ορίζεται όπως παρουσιάζεται πιο κάτω. Δεδομένου ενός συνόλου από εικονικές μηχανές VM= $\{vm_1,vm_2,\ldots,vm_n\}$ και ενός συνόλου από φυσικές μηχανές PM=$\{pm_1,pm_2,..,pm_m\}$. Κάθε vmi είναι μια τριάδα από: vm_i= cpu_i, ram_i, bw_i, i<=n και τα cpu_i, ram_i, bw_i συμβολίζουν τις απαιτήσεις της εικονική μηχανής i σε

cpu, μνήμη και εύρος ζώνης αντίστοιχα. Κάθε pm_j είναι μια τριάδα από: $pm_j = cpu_j, ram_j, bw_j$, $j<=m$ και τα cpu_j, ram_j, bw_j συμβολίζουν την ικανότητα των πόρων της φυσικής μηχανής j σε cpu, μνήμη και εύρος ζώνη αντίστοιχα. Επιπλέον x_{ij} ,$1<=i<=m$ και $1<=j<=n$ και y_i $1<=i<=m$ είναι μεταβλητές απόφασης και $x_{ij}=1$ εάν και μόνο αν vm_j φιλοξενείται πάνω στην pm_i και $y_i=1$ εάν η pm_i χρησιμοποιείται για να φιλοξενεί κάποια εικονική μηχανή. Ο στόχος είναι να ελαχιστοποιήσουμε το άθροισμα $\sum y_i$,$1<= i <=m$, ενώ ταυτόχρονα να βρεθούν όλες οι τιμές για το x_{ij}.[15]

Υπάρχουν πολλοί διαφορετικοί, έμμεσοι περιορισμοί με βάση την πιο πάνω περιγραφή. Συγκεκριμένα, πρώτος περιορισμός είναι ότι κάθε εικονική μηχανή μπορεί να φιλοξενείται σε μια μόνο φυσική μηχανή. Δεύτερος περιορισμός είναι ότι για κάθε είδος πόρου, τα ποσά αιτήσεων πόρων των εικονικών μηχανών που μοιράζονται την ίδια φυσική μηχανή πρέπει να είναι μικρότερα ή ίσα από την ικανότητα της φυσικής μηχανής που τις φιλοξενεί. Τελευταίος περιορισμός είναι ότι ο αριθμός των φυσικών μηχανών που φιλοξενούν εικονικές μηχανές δεν είναι περισσότερος από το m, $\sum y_i<=m$ (m αντιστοιχεί στο συνολικό αριθμό φυσικών μηχανών). [15]

Ικανοποίηση περιορισμών για το πρόβλημα τοποθέτησης εικονικών μηχανών

Σε αυτό το σημείο, θα περιγραφεί ο τρόπος αντιμετώπισης του προβλήματος τοποθέτησης εικονικών μηχανών χρησιμοποιώντας τις τεχνικές ικανοποίησης περιορισμών. Η βασική ιδέα του προγραμματισμού για ικανοποίηση περιορισμών είναι ότι ο χρήστης διατυπώνει το πραγματικό στο κόσμο πρόβλημα ως πρόβλημα ικανοποίησης περιορισμών(CSP) και έτσι ο βασικός σκοπός ενός επιλυτή περιορισμών είναι να υπολογίσει την λύση για το πρόβλημα αυτό. Όπως έχουμε ήδη αναφέρει στο κεφάλαιο 3, ένα CSP πρόβλημα είναι μια τριάδα, από μεταβλητές, πεδία τιμών των μεταβλητών και τους περιορισμούς πάνω σε αυτές τις μεταβλητές.

Έτσι, συνδυάζοντας τον ορισμό που δόθηκε πιο πάνω για το πρόβλημα τοποθέτησης εικονικών μηχανών και την βασική ιδέα για τα προβλήματα ικανοποίησης περιορισμών, το πρόβλημα της τοποθέτησης εικονικών μηχανών μπορεί να μοντελοποιηθεί ως πρόβλημα ικανοποίησης περιορισμών (CSP) με τον πιο κάτω τρόπο:[15]

Μεταβλητές

m - συνολικός αριθμός φυσικών μηχανών, n - συνολικός αριθμός εικονικών μηχανών

- $X=\{x_{ij} \mid i \in [1,m], j \in [1,n]\}$συμβολίζει την λύση στο πρόβλημα τοποθέτησης vm
- $Y=\{ yi \mid i \in [1,m]\}$, εάν η φυσική μηχανή pm_i χρησιμοποιείται τότε $y_i=1$.
- $Load\_Cpu=\{load\_cpu_i \mid i \in [1,m]\}$, load_cpu συμβολίζει το σύνολο των cpu αιτημάτων των εικονικών μηχανών που φιλοξενούνται στην φυσική μηχανή pm_i
- $Load\_Ram =\{load\_ram_i \mid i \in [1,m]\}$, load_ram συμβολίζει το σύνολο των αιτημάτων μνήμης των εικονικών μηχανών που φιλοξενούνται στην φυσική μηχανή pm_i .
- $Load\_BW =\{load\_bw_i \mid i \in [1,m]\}$, load_bw συμβολίζει το σύνολο των αιτημάτων εύρου ζώνης (bandwidth) των εικονικών μηχανών που φιλοξενούνται στην φυσική μηχανή pm_i .
- SolutionNum συμβολίζει τον αριθμό των φυσικών μηχανών που χρησιμοποιούνται στην λύση για το πρόβλημα τοποθέτησης εικονικών μηχανών.

Πεδία τιμών

- Για κάθε $x_{ij} \in X$, $x_{ij} \in [0,1]$
- Για κάθε $y_j \in Y, y_j \in [0,1]$
- Για κάθε $load\_cpu_i \in Load\_Cpu$, $load\_cpu_i \in [0, pm_{i,} cpu_i]$
- Για κάθε $load\_ram_i \in Load\_Ram$, $load\_ram_i \in [0, pm_{i,} ram_i]$
- Για κάθε $load\_bw_i \in Load\_BW$, $load\_bw_i \in [0, pm_{i,} bw_i]$
- SolutionNum $\in [0,m]$

Περιορισμοί

- C1: Το σύνολο των αιτημάτων των πόρων των εικονικών μηχανών που μοιράζονται την ίδια φυσική μηχανή πρέπει να είναι μικρότεροι ή ίσοι με τη χωρητικότητα των πόρων των φυσικών μηχάνημα που τις φιλοξενεί.
    1. Για κάθε i, $\sum_{j=1}^{n} xij * vm_j.cpu_j <= pm_i.cpu_i$,i $\in [1,m]$
    2. Για κάθε i, $\sum_{j=1}^{n} xij * vm_j.ram_j <= pm_i.ram_i$,i $\in [1,m]$
    3. Για κάθε i, $\sum_{j=1}^{n} xij * vm_j.bw_j <= pm_i.bw_i$,i $\in [1,m]$
- C2: $y_j=1$ εάν υπάρχει κάποια εικονική μηχανή που την φιλοξενεί η φυσική μηχανή j

  Για κάθε $y_j \in Y$, $y_j =1 \iff load\_cpu_j > 0$
- C3: Κάθε εικονική μηχανή μπορεί να τοποθετηθεί μόνο σε μια φυσική μηχανή

  Για κάθε j, $\sum_{j=1}^{n} xij =1$, j $\in [1,n]$

- C4: Ο αριθμός των φυσικών μηχανών που φιλοξενούν εικονικές μηχανές δεν πρέπει να είναι περισσότερος από το m, το συνολικό αριθμό φυσικών μηχανών.

SolutionNum = $\sum_{i=1}^{m} yi$<= m

4.2.2 Πρόβλημα Τοποθέτησης σε Κάδους (Bin Packing)

Το πρόβλημα «συσκευασίας κάδου» - bin backing είναι ένα συνδυαστικό NP-hard πρόβλημα. Σε αυτού του είδος την διατύπωση αντικείμενα διαφορετικών όγκων πρέπει να συσκευάζονται σε ένα πεπερασμένο αριθμό κάδων χωρητικότητας V με τέτοιο τρόπο ώστε να ελαχιστοποιείται ο αριθμός των κάδων που χρησιμοποιούνται. Οι περισσότεροι από τους πιο αποδοτικούς αλγορίθμους «συσκευασίας κάδου» χρησιμοποιούν ευρετικά για να φτάσουν σε λύση. Αυτό παρέχει μια λύση, η οποία αν και πολύ καλή στις περισσότερες περιπτώσεις μπορεί να μην είναι η βέλτιστη λύση.[1]

Το πρόβλημα της τοποθέτησης εικονικών μηχανών μπορεί να σχεδιαστεί ως ένα δοχείο συσκευασίας όπου οι φυσικές μηχανές μπορούν να θεωρηθούν ως κάδοι και οι εικονικές μηχανές μπορούν να θεωρηθούν ως αντικείμενα τα οποία πρέπει τοποθετηθούν στον κάδο.

4.2.3 Στοχαστικός Ακέραιος Προγραμματισμός

Η προσέγγιση Στοχαστικού Ακέραιου Προγραμματισμού χρησιμοποιείται για μοντελοποίηση προβλημάτων βελτιστοποίησης που εμπεριέχουν αβεβαιότητα. Τα περισσότερα από τα προβλήματα του πραγματικού κόσμου περιλαμβάνουν κάποιες άγνωστες παραμέτρους. Αυτό δεν μπορεί να συλληφθεί με απλό ακέραιο προγραμματισμό. Στοχαστικά μοντέλα προγραμματισμού μπορούν να επωφεληθούν από το γεγονός ότι οι κατανομές πιθανοτήτων που διέπουν τα δεδομένα είναι γνωστές ή μπορούν να εκτιμηθούν. Ο στόχος είναι να εντοπιστεί κάποια πολιτική η οποία να είναι εφικτή για όλες (ή σχεδόν όλες) τις πιθανές περιπτώσεις δεδομένων και να μεγιστοποιεί την προσδοκία κάποιας λειτουργίας των αποφάσεων και των τυχαίων μεταβλητών.[1]

Το πρόβλημα της τοποθέτησης εικονικών μηχανών μπορεί να σχεδιαστεί ως ένα στοχαστικό πρόβλημα ακέραιου προγραμματισμού ως εξής. Οι απαιτήσεις και οι τιμές είναι γνωστές (ή μπορούν να εκτιμηθούν). Οι απαιτήσεις και οι τιμές αναφέρονται στους πόρους(cpu/ram) των εικονικών μηχανών και των φυσικών μηχανών και ο στόχος είναι να ελαχιστοποιηθεί ο

προϋπολογισμός του χρήστη, δηλαδή να ελαχιστοποιηθεί ο αριθμός των φυσικών μηχανών έτσι ώστε να ανατεθούν όλες οι εικονικές μηχανές σε κάποια φυσική μηχανή.

4.2.4 Γενετικός Αλγόριθμος

Ένας γενετικός αλγόριθμος (GA) είναι μια τεχνική αναζήτησης η οποία χρησιμοποιείται για να βρεθούν ακριβής ή κατά προσέγγιση λύσεις για τη βελτιστοποίηση του προβλήματος αναζήτησης. Οι γενετικοί αλγόριθμοι κατηγοριοποιούνται ως αλγόριθμοι αναζήτησης με ευρετικά. Είναι εμπνευσμένοι από την εξελικτική βιολογία όπως είναι η κληρονομικότητα, η μετάλλαξη και η επιλογή. Ένας τυπικός γενετικός αλγόριθμος απαιτεί:

1. Μια γενετική αναπαράσταση του πεδίου τιμών της λύσης και
2. Μια συνάρτηση καταλληλότητας για την αξιολόγηση του πεδίου τιμών της λύσης.

Τυπικά ένας γενετικός αλγόριθμος ξεκινά με ένα σύνολο λύσεων, τα γονιδιώματα και εφαρμόζει γενετικούς τελεστές όπως η επιλογή, η διασταύρωση και η μετάλλαξη και έτσι οδηγείται σε μία βέλτιστη λύση.[1]

Το πρόβλημα της τοποθέτησης εικονικών μηχανών μπορεί να σχεδιαστεί ως ένα γενετικό πρόβλημα προγραμματισμού ως εξής. Το πεδίο τιμών της λύσης μπορεί να αναπαρασταθεί ως οι φυσικές μηχανές με ικανότητα τροφοδότησης των πόρων. Η συνάρτηση καταλληλότητας ορίζεται ως ο αριθμός των φυσικών μηχανών στην λύση. Ο στόχος είναι να δοθεί μια λύση που είναι κοντά στη βέλτιστη από την άποψη ελαχιστοποίησης του αριθμού των φυσικών μηχανών που χρησιμοποιούνται.

4.2.5 Σύγκριση Αλγορίθμων – Ποιος δουλεύει Καλύτερα;

Ο **Προγραμματισμός με Περιορισμούς** είναι χρήσιμος σε περιπτώσεις όπου έχουμε τα δεδομένα εισόδου εξαρχής, δηλαδή πριν υπολογιστούν οι συναρτήσεις κόστους γνωρίζουμε τις απαιτήσεις των εικονικών μηχανών. Οι τεχνικές που χρησιμοποιούν προγραμματισμού με περιορισμούς είναι εύκολα επεκτάσιμες ώστε να λάβουν υπόψη πρόσθετους περιορισμούς. Είναι εύκολο να καθοριστούν περιορισμοί έτσι ώστε να χειριστούν τους συνδυασμούς μεταξύ πολλαπλών στόχων που θα μπορούσαν να έρχονται σε σύγκρουση ο ένας με τον άλλο. Ο χρόνος που απαιτείται για να δημιουργηθεί μια βέλτιστη λύση είναι υψηλός καθώς ο αριθμός των περιορισμών αυξάνει.

Η προσέγγιση **Τοποθέτησης σε Κάδους (Bin Packing),** μπορεί να θεωρηθεί ως υποσύνολο της προσέγγισης προγραμματισμού με Περιορισμούς. Η προσέγγιση αυτή είναι χρήσιμη για τη δυναμική τοποθέτηση εικονικών μηχανών, ιδίως όταν η ζήτηση είναι εξαιρετικά μεταβλητή. Πρόκειται για μια προσέγγιση που βασίζεται σε ευρετικά. Έτσι είναι προφανές ότι δεν μπορεί να δώσει μια βέλτιστη λύση. Ωστόσο, αυτό θα δημιουργήσει πάντα μια καλή λύση σε μεγάλο χρονικό διάστημα. Μια προσέγγιση αυτού του είδους είναι πραγματικά χρήσιμη όταν όλες οι φυσικές μηχανές έχουν την ίδια ποσότητα μνήμης και δυνατότητα επεξεργασίας.

Η προσέγγιση **Στοχαστικού Ακέραιου Προγραμματισμού** είναι χρήσιμη στις περιπτώσεις κατά τις οποίες δεν είναι γνωστές οι μελλοντικές απαιτήσεις και οι τιμές των πόρων, αλλά οι κατανομές πιθανοτήτων τους είτε είναι γνωστές ή μπορούν να υπολογιστούν. Αυτή είναι η καλύτερη τεχνική που θα χρησιμοποιηθεί στην περίπτωση όπου υπάρχουν δύο ή περισσότερες αβέβαιες παράμετροι από τις οποίες εξαρτάται το κόστος.

Ο **Γενετικός Αλγόριθμος** είναι ένας τρόπος για να λυθεί το Πρόβλημα Τοποθέτησης σε Κάδους με ορισμένους περιορισμούς. Η προσέγγιση αυτή απαιτεί περισσότερο υπολογιστικό χρόνο και υψηλότερους υπολογιστικούς πόρους σε σύγκριση με την προσέγγιση τοποθέτησης σε κάδους. Είναι ιδιαίτερα χρήσιμος για τις στατικές τοποθετήσεις, που είναι σε σενάρια όπου οι απαιτήσεις δεν διαφέρουν πάνω από ένα σημαντικό χρονικό διάστημα.[1]

Κάθε ένας από τους πιο πάνω αλγόριθμους τοποθέτησης εικονικών μηχανών λειτουργεί καλά υπό συγκεκριμένες προϋποθέσεις. Έτσι, είναι σημαντικό να επιλεχθεί μια τεχνική που ταιριάζει στις ανάγκες του χρήστη και του παρόχου του υπολογιστικού νέφους.

Στην παρούσα πτυχιακή εργασία θα ασχοληθούμε με την υλοποίηση της προσέγγισης προγραμματισμού με περιορισμούς όπου η θεωρία και η ανάλυση των περιορισμών αναλύεται στο επόμενο κεφάλαιο.

4.3 Τροφοδότηση των Πόρων με περιορισμούς προϋπολογισμού για προσαρμοστικές εφαρμογές στο Υπολογιστικό Νέφος

Στο υπο-κεφαλαίου αυτού, θα γίνει λεπτομερής αναφορά σε μια προσέγγιση του προβλήματος τροφοδότησης με χρήση προγραμματισμού με περιορισμούς. Θα επικεντρωθούμε στην αυτόνομη και δυναμική κατανομή των πόρων που συνδέονται με την εκτέλεση μιας πρωτοεμφανιζόμενης κλάσης εφαρμογών, τις προσαρμοστικές εφαρμογές(adaptive applications) στα περιβάλλοντα του υπολογιστικού νέφους. Συγκεκριμένα, θεωρούμε ότι υπάρχει ένα σταθερό όριο χρόνου εκτέλεσης καθώς και ένας σταθερός προϋπολογισμός για τους πόρους. Με αυτούς τους περιορισμούς, μια προσαρμοστική εφαρμογή χρειάζεται να μεγιστοποιεί την μετρική ποιότητας της υπηρεσίας(Qos) και συγκεκριμένα την τιμή της συνάρτησης οφέλους(benefit function), η οποία αναφέρεται στο τι είναι πιο επιθυμητό να υπολογιστεί εντός της προθεσμίας χρόνου εκτέλεσης (time-limit).[12]

Ο υπολογισμός χρησιμότητας(utility computing) αποτελούσε όραμα το οποίο είχε διατυπωθεί πριν από 40 χρόνια περίπου. Ο υπολογισμός χρησιμότητας αναφέρεται στην ιδέα ότι υπολογιστικοί πόροι και υπηρεσίες μπορούν να παραδοθούν, να χρησιμοποιηθούν και καταβάλλονται όπως το νερό ή και τον ηλεκτρισμό. Αυτό το όραμα πραγματοποιείται με την εμφάνιση του υπολογιστικού νέφους.[12]

Η δυναμική τροφοδότηση υπολογιστικών και αποθηκευτικών πόρων είναι εφικτή στο υπολογιστικό νέφος. Στόχος είναι να κρατηθεί ο προϋπολογισμός(recourse Budget) των πόρων στο ελάχιστο ενώ παράλληλα πληρούνται οι ανάγκες των εφαρμογών. Η κατανομή των πόρων είναι επιθυμητή σε περιβάλλοντα υπολογιστικού νέφους και μπορεί να πραγματοποιηθεί αυτόματα και δυναμικά με βάση τις υψηλού ανάγκες των χρηστών, όπως για παράδειγμα προϋπολογισμός των πόρων, περιορισμός στον χρόνο εκτέλεσης ή και επιθυμητή ποιότητα αποτελεσμάτων.

Το πρόβλημα τροφοδότησης των πόρων γίνεται ιδιαίτερη πρόκληση για μια πρωτοεμφανιζόμενη κλάση εφαρμογών οι οποίες αναφέρονται ως προσαρμοστικές εφαρμογές(adaptive applications). Σ’ αυτές τις εφαρμογές υπάρχει μια ειδική ευελιξία στον υπολογισμό. Η ευελιξία αυτή αναφέρεται στο γεγονός ότι δεδομένα εισόδου μπορούν να δοκιμαστούν με διαφορετικούς ρυθμούς, μοντέλα μπορούν να εκτελούνται με διαφορετική

χρονική και χωρική λεπτομέρεια και βαθμό ανάλυσης ή με διαφορετικές τιμές για τον συντελεστή σφάλματος.[12]

Στόχος μιας προσαρμοστικής εφαρμογής είναι να περιλαμβάνει μια σειρά από στοιχεία υπηρεσιών που αλληλεπιδρούν και συμβολίζονται ως S1,S2,…,Sn. Μια εφαρμογή έχει την συνάρτηση οφέλους της(benefit function) η οποία συμβολίζεται ως B. Αυτή η συνάρτηση λαμβάνει ορισμένες παραμέτρους εφαρμογής ως είσοδο και εξάγει την ποιότητα της υπηρεσίας (Qos) η οποία ονομάζεται όφελος(benefit). Η συνάρτηση οφέλους είναι συγκεκριμένη και ορίζεται από τον χρήστη. Σε αυτές τις περιπτώσεις προβλημάτων υπάρχει ένας προκαθορισμένος περιορισμός χρόνου εκτέλεσης ο οποίος συμβολίζεται με Tc καθώς επίσης υπάρχει και προκαθορισμένος προϋπολογισμός των πόρων για κάθε εφαρμογή ο οποίος συμβολίζεται με Co.

Στόχος είναι να εκτελεστεί η διαδικασία έτσι ώστε να μεγιστοποιηθεί το όφελος της εφαρμογής(benefit value) εντός τις προθεσμίας χρόνου Tc και του προϋπολογισμού των πόρων Co. Οι προσαρμοστικές παράμετροι επηρεάζουν σε μεγάλο βαθμό το όφελος το οποίο λαμβάνεται από μια εκτέλεση. Κάθε υπηρεσία είναι δυνατόν να έχει μια ή περισσότερες από τέτοιες προσαρμοστικές παραμέτρους. Ορισμένες επιλογές στις τιμές αυτών των παραμέτρων είναι πιθανόν να επιταχύνουν την διαδικασία ή να απαιτούν λιγότερους πόρους μνήμης ή Cpu. Έτσι η προκύπτουσα τιμή για το όφελος θα είναι χαμηλότερη. Ακόμη, άλλες επιλογές των τιμών αυτών πιθανόν να οδηγήσουν στην αύξηση του οφέλους.[12]

Έτσι, σκοπός είναι η επιλογή ορθών συνδυασμών μεταξύ του οφέλους και του κόστους των πόρων, μέσω της προσαρμογής των τιμών των παραμέτρων. Όταν οι τιμές των παραμέτρων αυτών τροποποιούνται, πιθανόν να χρειαστεί να αλλάξει η κατανομή σε cpu/μνήμη έτσι ώστε να διατηρηθεί το επιθυμητό ποσοστό προόδου.

Κεφάλαιο 5

Αυτόνομη Εικονική Διαχείριση Πόρων

5.1 Εισαγωγή

Οι πλατφόρμες του υπολογιστικού νέφους φιλοξενούν διάφορες ανεξάρτητες εφαρμογές σε έναν κοινόχρηστο χώρο συγκέντρωσης πόρων με την ικανότητα κατανομής υπολογιστικής ισχύς σε εφαρμογές. Η χρήση των τεχνικών εικονοποίησης παρέχουν μεγάλη ευελιξία ως προς την ικανότητα ανάθεσης πολλών εικονικών μηχανών στην ίδια φυσική μηχανή, ως προς την αλλαγή μεγέθους της χωρητικότητας των εικονικών μηχανών και τέλος ως προς την μετανάστευση εικονικών μηχανών στις φυσικές μηχανές.

Οι υποδομές υπολογιστικού νέφους θα πρέπει να βελτιώσουν τα ποσοστά χρησιμοποίησης των πόρων. Η τεχνολογία των συστημάτων υπολογιστικού νέφους είναι η εικονοποίηση διακομιστή η οποία επιτρέπει την αποσύνδεση εφαρμογών και υπηρεσιών από τις υποδομές φυσικών εξυπηρετητών. Όπως έχουμε προαναφέρει, με τον όρο εικονοποίηση διακομιστή εννοούμε την ταυτόχρονη εκτέλεση πολλών εικονικών μηχανών στην κορυφή μιας φυσικής μηχανής.

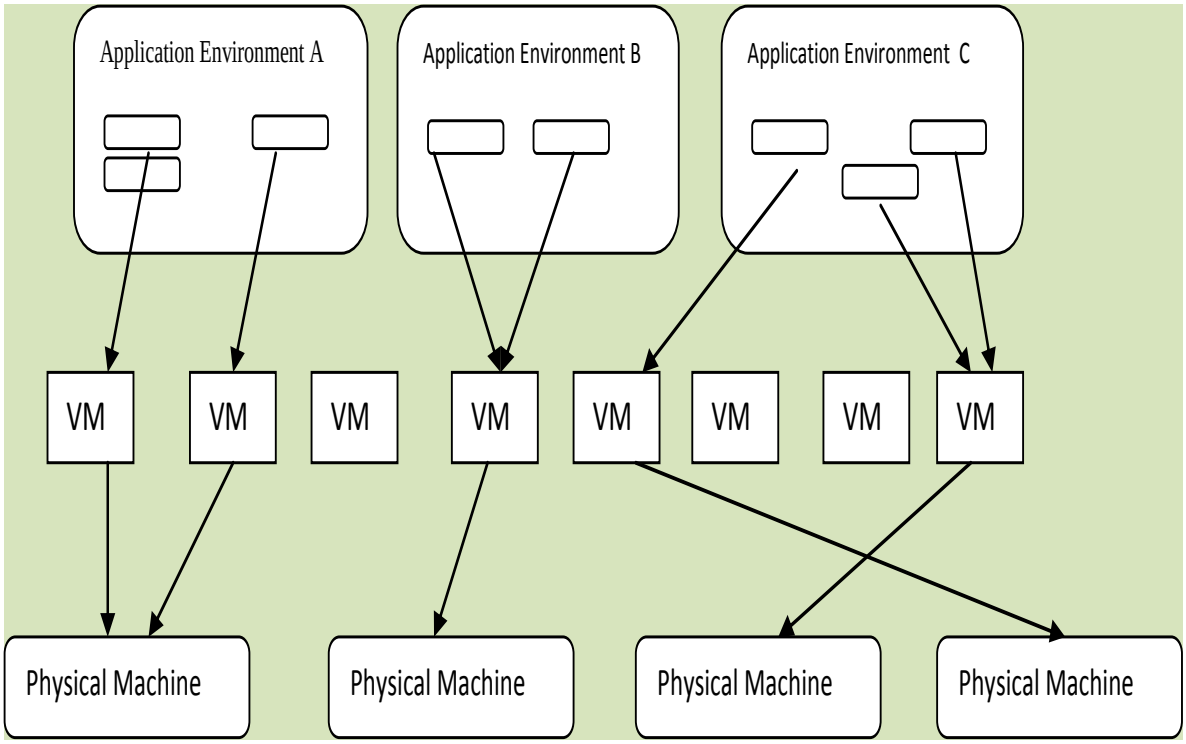
Η βασική πρόκληση για τους παρόχους του υπολογιστικού νέφους είναι η αυτοματοποίηση της διαχείρισης των εικονικών μηχανών, λαμβάνοντας υπόψη τις υψηλού απαιτήσεις των εφαρμογών (Qos) όπως επίσης και το κόστος διαχείρισης των πόρων.

Απώτερος σκοπός είναι η αυτόνομη διαχείριση των πόρων, έτσι που να αποσυνδέεται η τροφοδότηση των πόρων από την δυναμική τοποθέτηση των εικονικών μηχανών στις φυσικές μηχανές. Αυτή η αυτόνομη διαχείριση στοχεύει στην βελτιστοποίηση της ικανοποίησης των εφαρμογών(global utility function-UGlobal) όπως επίσης και στην δυνατότητα αυτοματοποίησης δυναμικών τροφοδοτήσεων και τοποθετήσεων των εικονικών μηχανών. Η συνάρτηση ικανοποίησης(UGlobal) απεικονίζεται με μια ακέραια τιμή η οποία προσδιορίζει την ικανοποίηση της κάθε εφαρμογής με συγκεκριμένη κατανομή πόρων(CPU, RAM) σε σχέση με τους στόχους επίδοσης.[5]

Τα δύο επίπεδα τα οποία πρέπει να διαχειρίζονται(Εικόνα 5.1) είναι το επίπεδο τροφοδότησης εικονικών μηχανών και το επίπεδο τοποθέτησης εικονικών μηχανών. Το πρώτο επίπεδο είναι υπεύθυνο για την κατανομή της χωρητικότητας των πόρων με την μορφή εικονικές μηχανές σε εφαρμογές και το δεύτερο επίπεδο είναι υπεύθυνο για την ανάθεση των εικονικών μηχανών σε φυσικές μηχανές.

Τι είναι το Qos?

Το Quality of service ή αλλιώς ποιότητα της υπηρεσίας είναι η δυνατότητα παροχής διαφορετικής προτεραιότητας σε διαφορετικές εφαρμογές, χρήστες ή ροές δεδομένων ή η εγγύηση ενός ορισμένου επιπέδου απόδοσης.



Εικόνα 5.1: Δυναμική Κατανομή Πόρων

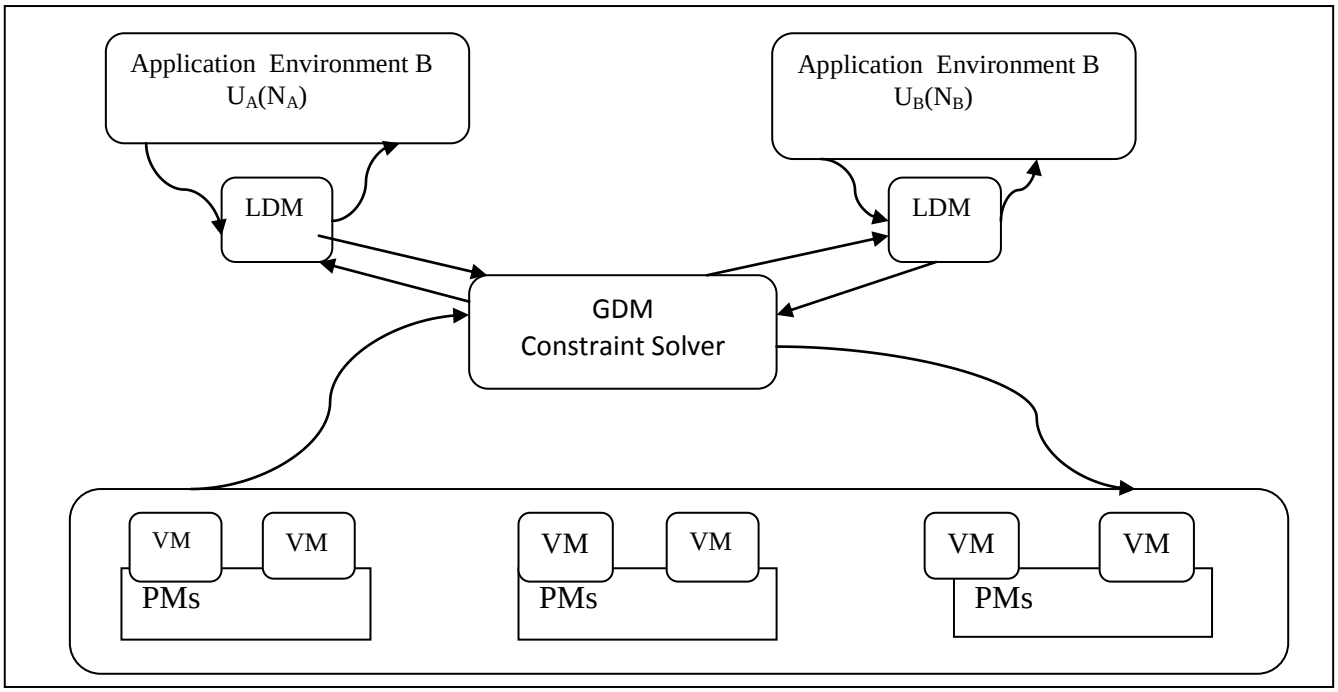
5.2 Αρχιτεκτονική του Συστήματος

Η αρχιτεκτονική του Συστήματος παρουσιάζεται στην Εικόνα 5.2(1). Το κέντρο δεδομένων(datacenter) αποτελείται από ένα σύνολο φυσικών μηχανών και η κάθε μια φιλοξενεί πολλές εικονικές μηχανές. Η αρχιτεκτονική του συστήματος αποτελείται από το LDM(local decision module), το AE(application environment) και το GDM(global decision module).[5]

Το AE ενθυλακώνει μια εφαρμογή που φιλοξενείται από το σύστημα υπολογιστικού νέφους. Ένα AE συνδέεται με συγκεκριμένους στόχους επίδοσης οι οποίοι καθορίζονται από την σύμβαση SLA. Η σύμβαση SLA είναι μια συμφωνία μεταξύ δύο ή περισσότερων μέρων, όπου το ένα είναι ο πελάτης και οι άλλοι είναι πάροχοι υπηρεσιών. Ένα τρέξιμο εικονικής μηχανής συνδέεται μόνο με ένα AE. Οι εικονικές μηχανές στην διάθεση των εφαρμογών, πρέπει να επιλεχθούν ανάμεσα σε ένα σύνολο προκαθορισμένων κατηγοριών εικονικών μηχανών. Κάθε κατηγορία εικονικής μηχανής έχει συγκεκριμένη ικανότητα σε CPU(σε Ghz) και μνήμη RAM (σε Gb).

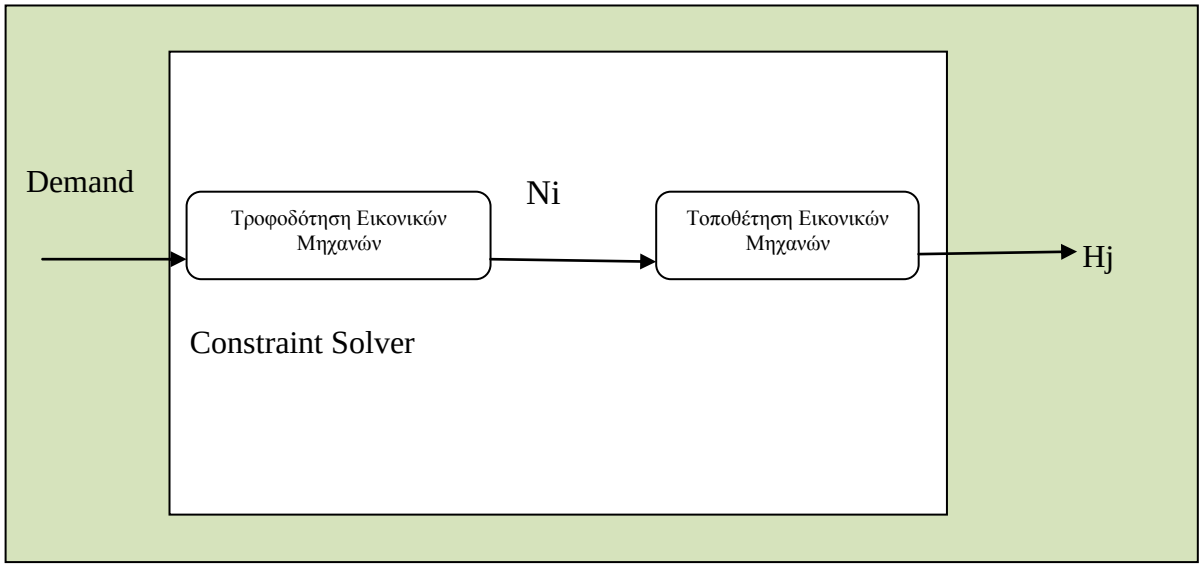
Το LDM σχετίζεται με κάθε AE. Σκοπός του LDM είναι ο υπολογισμός της συνάρτησης ικανοποίησης(UGlobal) η οποία δίνει ένα μέτρο ικανοποίησης της εφαρμογής με συγκεκριμένη κατανομή πόρων(CPU, RAM). Κάθε LDM πραγματοποιεί την κατανομή περισσότερων εικονικών μηχανών ή απελευθέρωση υπαρχόντων εικονικών μηχανών προς το AE. Τα LDMs αλληλεπιδρούν με το global Decision Module(GDM).

Το GDM είναι υπεύθυνο για την διαιτησία των απαιτούμενων πόρων που προέρχονται από κάθε AE και αντιμετωπίζει κάθε LDM ως ένα μαύρο κουτί χωρίς να γνωρίζει την φύση της εφαρμογής ή τον τρόπο που το LDM υπολογίζει την συνάρτηση ικανοποίησης. Είναι υπεύθυνο για δύο στόχους. Πρώτος του στόχος είναι ο καθορισμός του διανύσματος κατανομής Ni των εικονικών μηχανών για κάθε εφαρμογή αi(VM provisioning) και δεύτερος του στόχος είναι η διάθεση αυτών των εικονικών μηχανών σε φυσικές μηχανές προκειμένου να ελαχιστοποιηθεί ο αριθμός των ενεργών φυσικών μηχανών(VM placement). Το GDM παίρνει ως είσοδο την συνάρτηση ικανοποίησης από κάθε LDM. Τέλος ειδοποιά το LDM ότι μια καινούργια εικονική μηχανή με συγκεκριμένη χωρητικότητα έχει διατεθεί για την εφαρμογή και ότι μια υπάρχουσα εικονική μηχανή έχει αναβαθμιστεί ή υποβαθμιστεί(όπως για παράδειγμα να έχει αλλάξει η χωρητικότητα των πόρων και η κατηγορία).[5]



Εικόνα 5.2(1) Αρχιτεκτονική συστήματος

Οι δύο φάσεις του προβλήματος, τροφοδότησης(vm provisioning) και τοποθέτησης(vm placement) εικονικών μηχανών, εκφράζονται και οι δύο ως προβλήματα ικανοποίησης περιορισμών(Εικόνα 5.2(2)) και όπως θα δούμε πιο κάτω διεκπεραιώνονται από την επίλυση περιορισμών χρησιμοποιώντας κάποιους συγκεκριμένους επιλυτές.



Εικόνα 5.2(2): Constraint Solving : Οι δυο φάσεις του προβλήματος

5.3 Φάση Τροφοδότησης Εικονικών Μηχανών - Virtual Machine Provisioning Phase

Όπως αναφέραμε και πιο πάνω, στόχος της φάσης αυτής είναι η εύρεση του διανύσματος κατανομής εικονικών μηχανών N_i για κάθε εφαρμογή μεγιστοποιώντας την συνάρτηση ικανοποίησης. Πιο κάτω παρουσιάζονται οι παράμετροι του προβλήματος για την επισημοποίηση των λειτουργιών. Συγκεκριμένα, $A=(\alpha_1,\alpha_2,\alpha_i,..,\alpha_m)$ δηλώνει το σύνολο των εφαρμογών και $P=(p_1,p_2,p_j,..,p_q)$ δηλώνει το σύνολο των φυσικών μηχανών. Κάθε κατηγορία εικονικής μηχανής έχει συγκεκριμένη ικανότητα σε CPU(σε Ghz) και μνήμη Ram(σε Gb). Υπάρχουν c κατηγορίες εικονικών μηχανών, οι οποίες προσδιορίζονται από το διάνυσμα $S=(S_1,S_2,S_k..,S_c)$, όπου $S_k=(S_k^{Cpu}, S_k^{Ram})$, S_k^{Cpu} προσδιορίζει την ικανότητα της Cpu και S_k^{Ram} προσδιορίζει την ικανότητα μνήμης Ram της εικονικής μηχανής της κατηγορίας k. Τα διανύσματα C_j^{cpu} και C_j^{ram} προσδιορίζουν την ικανότητα της φυσικής μηχανής p_j σε Cpu και σε μνήμη Ram αντίστοιχα.

Το N_i είναι το διάνυσμα κατανομής εικονικών μηχανών της εφαρμογής α_i όπου, $N_i=(n_{i1},n_{i2},\ldots,n_{ik},n_{im})$ - n_{ik} είναι ο αριθμός των εικονικών μηχανών της κατηγορίας S_k που αποδίδεται στην εφαρμογή α_i. Η συνάρτηση για το επίπεδο των πόρων(recource level function) U_i για την εφαρμογή α_i ορίζεται ως $U_i = f_i(N_i)= c1_{i1}*n_{i1} + c1_{i2}*n_{i2} + ... + c1_{im}*n_{im}$. Η πιο πάνω συνάρτηση εκφράζει το συνολικό επίπεδο των πόρων που θα έχουν οι εφαρμογές ανάλογα με τον συντελεστή/βαρύτητα($c1_{im}$) που θα αποδοθεί σε κάθε εφαρμογή για κάθε κατηγορία εικονικών μηχανών. Το λειτουργικό κόστος για την εφαρμογή α_i ορίζεται ως $Cost_i = c2_{i1}*n_{i1} + c2_{i2}*n_{i2} + ... + c2_{im}*n_{im}$. Η εξίσωση αυτή αντιπροσωπεύει το συνολικό λειτουργικό κόστος που θα έχουν οι εφαρμογές ανάλογα με τον συντελεστή/βαρύτητα($c2_{im}$) που θα αποδοθεί σε κάθε εφαρμογή για κάθε κατηγορία εικονικών μηχανών. Άνω φράγμα για τον αριθμό των εικονικών μηχανών σε κάθε κατηγορία είναι το διάνυσμα $N_i^{max}=(n_{i1}^{max},n_{i2}^{max},n_{ik}^{max},n_{im}^{max})$. Ο συνολικός αριθμός εικονικών μηχανών για κάθε εφαρμογή εκφράζεται από το διάνυσμα $T_i^{max.}$.

Στην παρούσα διατύπωση του προβλήματος υποθέτουμε ότι οι εφαρμογές έχουν σταθερό φορτίο, υπό την έννοια ότι οι απαιτήσεις αυτές δεν αυξομειώνονται. Συγκεκριμένα, οι απαιτήσεις των εφαρμογών σε πόρους παραμένουν σταθερές κατά την διάρκεια εκτέλεσης τους.

Οι βασικοί περιορισμοί αυτής της φάσης φαίνονται πιο κάτω

1. $n_{ik} \leq n_{ik}^{max}$ $\quad 1 \leq i \leq m$ και $1 \leq k \leq c$

   Στον περιορισμό αυτό ο δείκτης i αναφέρεται στις εφαρμογές και ο k αναφέρεται στις κατηγορίες εικονικών μηχανών. Με αυτό τον περιορισμό εξασφαλίζεται ότι η ανάθεση που θα γίνει στο διάνυσμα N_i για κάθε εφαρμογή και για κάθε κατηγορία εικονικών μηχανών, δεν θα ξεπερνά το άνω φράγμα για τον αριθμό των εικονικών μηχανών για κάθε κατηγορία εικονικών μηχανών.

2. $\sum_{k=1}^{c} n_{ik} \leq T_i^{max}$ $\quad 1 \leq i \leq m$ και $1 \leq k \leq c$

   Στον περιορισμό αυτό ο δείκτης i αναφέρεται στις εφαρμογές και ο k αναφέρεται στις κατηγορίες εικονικών μηχανών. Το c αναφέρεται στο συνολικό αριθμό κατηγοριών εικονικών μηχανών. Με αυτόν το περιορισμό εξασφαλίζεται ότι το συνολικό άθροισμα των εικονικών μηχανών για κάθε εφαρμογή θα είναι μικρότερο ή ίσο του συνολικού αριθμού των εικονικών μηχανών που απαιτεί η συγκεκριμένη εφαρμογή.

3. $\sum_{i=1}^{m}\sum_{k=1}^{c} n_{ik} * S_k^{cpu} \leq \sum_{j=1}^{q} C_j^{cpu}$ και $\sum_{i=1}^{m}\sum_{k=1}^{c} n_{ik} * S_k^{ram} \leq \sum_{j=1}^{q} C_j^{ram}$

   Οι δύο πιο πάνω περιορισμοί, είναι παρόμοιοι αλλά ο ένας αναφέρεται στην Cpu και ο άλλος στην μνήμη Ram. Αυτοί οι περιορισμοί αναφέρουν ότι, το συνολικό άθροισμα για κάθε κατηγορία εικονικής μηχανής για την κατανομή που θα γίνει στο διάνυσμα N_i επί την ικανότητα της Cpu(ή Ram αντίστοιχα) της κατηγορίας αυτής εικονικών μηχανών, δεν θα ξεπερνά το συνολικό άθροισμα των ικανοτήτων των φυσικών μηχανών σε Cpu (ή Ram αντίστοιχα).

4. Όπως έχουμε αναφέρει και προηγουμένως σκοπός της φάσης αυτής είναι να γίνει μια κατανομή στο διάνυσμα N_i για κάθε εφαρμογή και να προσδιοριστεί πόσες εικονικές μηχανές από κάθε κατηγορία χρειάζεται η κάθε εφαρμογή. Αυτή η κατανομή θα πρέπει να γίνει με τέτοιο τρόπο έτσι ώστε να μεγιστοποιείται η συνάρτηση ικανοποίησης των εφαρμογών (Global Utility Function) η οποία δίνει ένα μέτρο ικανοποίησης της εφαρμογής με συγκεκριμένη κατανομή πόρων (Cpu,Ram). Η συνάρτηση αυτή εκφράζεται από την πιο κάτω εξίσωση :

   $$Uglobal = \text{maximize } (10-e) * \sum_{i=1}^{m}(a_i * U_i) - (e * total) * \sum_{i=1}^{m}(Cost_i)$$

   όπου ai είναι η βαρύτητα κάθε εφαρμογής, total είναι το άθροισμα όλων των βαρυτήτων των εφαρμογών και e συμβολίζει τον συντελεστής κόστους. Σκοπός της πιο πάνω εξίσωσης είναι να ισορροπηθεί η διαφορά ανάμεσα στο επίπεδο των

πόρων και στο λειτουργικό κόστος το οποίο θα έχει η κάθε εφαρμογή σε κάθε κατηγορία εικονικών μηχανών.

5.4 Φάση Τοποθέτησης Εικονικών Μηχανών στις Φυσικές Μηχανές - Virtual Machine Placement

Όπως έχει προαναφερθεί, στόχος της φάσης αυτής είναι η διάθεση αυτών των εικονικών μηχανών σε φυσικές μηχανές προκειμένου να ελαχιστοποιηθεί ο αριθμός των ενεργών φυσικών μηχανών(VM placement). Η φάση αυτή παίρνει ως είσοδο το διάνυσμα N_i από την φάση τροφοδότησης εικονικών μηχανών(Vm provisioning) και καταλήγει στο ενιαίο διάνυσμα $V=(vm_1,vm_2,vm_l,\ldots,vm_u)$, το οποίο απαριθμεί ποιες εικονικές μηχανές τρέχουν. Για κάθε φυσική μηχανή p_j που ανήκει στο διάνυσμα $P=(p_1,p_2,p_j,..,p_q)$, το διάνυσμα $H_j=(h_{j1},h_{j2},h_{jl},\ldots,h_{ju})$, συμβολίζει το σύνολο των εικονικών μηχανών που ανατίθενται στην φυσική μηχανή p_j. Τυπικά, η τοποθέτηση εικονικών μηχανών σε φυσικές μηχανές συμβολίζεται ως $H_{jl}=1$ εάν η φυσική μηχανή p_j φιλοξενεί την εικονική μηχανή vm_l. Το H_j είναι δυαδικό διάνυσμα(0 ή 1). Το διάνυσμα $R=(r_1,r_2,r_l,..,r_u)$ προσδιορίζει την ικανότητα των πόρων(Cpu και Ram) όλων των εικονικών μηχανών και συμβολίζεται με $r_l=(r_l^{cpu},r_l^{ram})$.

Οι βασικοί περιορισμοί αυτής της φάσης είναι:

1. $\sum_{l=1}^{u} r_l^{cpu} * h_{jl} \leq C_j^{cpu}$ και $\sum_{l=1}^{u} r_l^{ram} * h_{jl} \leq C_j^{ram}$ $\quad 1 \leq j < q$
   Οι δύο πιο πάνω περιορισμοί, είναι παρόμοιοι αλλά ο ένας αναφέρεται στην ικανότητα σε Cpu και ο άλλος στην μνήμη Ram. Οι δύο αυτοί περιορισμοί αναφέρουν ότι το συνολικό άθροισμα των Cpu των εικονικών μηχανών (αντίστοιχα για Ram), οι οποίες θα ανατεθούν σε μια(στην ίδια) φυσική μηχανή δεν θα ξεπερνά την Cpu (αντίστοιχα για Ram) αυτής της εικονικής μηχανής.
2. Όπως έχουμε αναφέρει και προηγουμένως σκοπός αυτής της φάσης είναι να πραγματοποιηθεί η διάθεση των εικονικών μηχανών σε φυσικές μηχανές προκειμένου να ελαχιστοποιηθεί ο αριθμός των ενεργών φυσικών μηχανών. Το διάνυσμα U αναφέρει κατά πόσο μια φυσική μηχανή είναι ενεργή ή όχι δηλαδή, αν τις έχουν ανατεθεί εικονικές μηχανές ή όχι. Έτσι σκοπός είναι να ελαχιστοποιηθεί το άθροισμα $\sum_{j=1}^{q} U_j$, όπου $U_j=1$ εάν η εικονική μηχανή vm_l ανήκει στο V και $h_{jl}=1$, αλλιώς ίσο με 0.

Κεφάλαιο 6

Υλοποίηση Προβλήματος Αυτόνομης Εικονικής Διαχείρισης Πόρων

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει πλήρης αναφορά στην υλοποίηση του προβλήματος αυτόνομης εικονικής Διαχείρισης Πόρων. Η αναφορά στην θεωρία γύρω από τον ορισμό του προβλήματος έχει γίνει στο κεφάλαιο 5. Έτσι, σε αυτό το κεφάλαιο θα γίνει αναφορά και ανάλυση των περιορισμών για κάθε μια από τις δύο φάσεις του προβλήματος, τροφοδότησης εικονικών μηχανών (vm provisioning) και τοποθέτησης εικονικών μηχανών (vm placement). Επιπλέον, θα παρουσιαστούν οι επιλυτές με τους οποίους έχει λυθεί το πρόβλημα, οι οποίοι είναι ο Minzinc, Minisat+ και ο Bsolo. Θα περιγραφεί ο τρόπος με τον οποίο υλοποιήθηκε το

πρόβλημα στον επιλυτή Minizinc και έπειτα πώς μεταφράστηκε το πρόβλημα για τους ψειδοδυαδικούς επιλυτές Minisat+ και Bsolo.

6.2 Γενικά για τα Εργαλεία Υλοποίησης

6.2.1 Η γλώσσα προγραμματισμού με περιορισμούς- Minizinc

Η Minizinc είναι μια γλώσσα η οποία σχεδιάστηκε για τον καθορισμό και επίλυση περιορισμών και για τη βελτιστοποίηση και την απόφαση προβλημάτων όσων αφορά ακέραιους και πραγματικούς αριθμούς. Ένα μοντέλο Minizinc δεν υπαγορεύει πώς να λυθεί το πρόβλημα. Η Minizinc επιτρέπει τον προσδιορισμό παγκόσμιων περιορισμών (global constraints) όπως είναι οι περιορισμοί, alldiferent, cumulative κλπ. Σε κάθε μοντέλο στην γλώσσα Minizinc θα πρέπει να καθορίζονται τα τρία σημεία που αναφέρθηκαν στο κεφάλαιο 3 τα οποία ορίζουν τα προβλήματα ικανοποίησης περιορισμών.[7]

Μέσα από ένα παράδειγμα, θα κατανοήσουμε καλύτερα την σύνταξη της γλώσσας Minizinc. Στο κεφάλαιο 3 παρουσιάστηκε το πρόβλημα χρωματισμού χάρτη. Το πρόβλημα αυτό περιλαμβάνει τον χρωματισμό διαφορετικών περιοχών ενός καθορισμένου χάρτη, με περιορισμένο αριθμό χρωμάτων συγκεκριμένα τρία χρώματα, και υπόκειται στον περιορισμό ότι δύο γειτονικές περιοχές δεν επιτρέπεται να έχουν το ίδιο χρώμα. Στην παρούσα διατύπωση ας θεωρήσουμε τον χάρτη της Αυστραλίας και ότι έχουμε τρία χρώματα κίτρινο , κόκκινο και μπλε.

Για κάθε μια από τις εφτά περιοχές ορίζουμε μία μεταβλητή. Στις μεταβλητές δίνουμε τα εξής ονόματα: *WA, NT, SA, Q ,NSW, V ,T* και σε κάθε μια από αυτές τις μεταβλητές πρέπει να γίνει ανάθεση ενός χρώματος. Κάθε μεταβλητή έχει ως πεδίο τιμών το σύνολο *{ κίτρινο, κόκκινο, μπλε}*. Ο περιορισμός ο οποίος αναγράφεται πιο κάτω αποτελεί εγγύηση για το γεγονός ότι κάθε ζεύγος γειτονικών περιοχών δεν μπορεί να έχει το ίδιο χρώμα:

$$WA \neq NT \wedge WA \neq SA \wedge NT \neq SA \wedge NT \neq Q \wedge SA \neq Q \wedge$$
$$SA \neq NSW \wedge SA \neq V \wedge Q \neq NSW \wedge NSW \neq V$$

Το πρόβλημα αυτό στην γλώσσα Minizinc μετατρέπεται όπως βλέπουμε πιο κάτω:

```
% Colouring Australia using nc colours
int: nc = 3;
var 1..nc: wa; var 1..nc: nt; var 1..nc: sa; var 1..nc: q;   Μεταβλητές και πεδία τιμών
var 1..nc: nsw; var 1..nc: v; var 1..nc: t;
constraint wa != nt;
constraint wa != sa;
constraint nt != sa;
constraint nt != q;                          Περιορισμοί Προβλήματος
constraint sa != q;
constraint sa != nsw;
constraint sa != v;
constraint q != nsw;
constraint nsw != v;
solve satisfy;  ------------------------>  Είδος Προβλήματος: Πρόβλημα Ικανοποίησης
output ["wa=", show(wa), "\t nt=", show(nt),
"\t sa=", show(sa), "\n", "q=", show(q),
"\t nsw=", show(nsw), "\t v=", show(v), "\n",
"t=", show(t), "\n"];
```

Με βάση την πιο πάνω διατύπωση, υπάρχουν 7 μεταβλητές όσες και οι περιοχές. Η κάθε μεταβλητή έχει πεδίο τιμών 1..3, όσος και ο δυνατός αριθμός χρωμάτων. Στο τέλος του μοντέλου πριν το τύπωμα της λύσης βλέπουμε την έκφραση «***solve*** *satisfy;*». Αυτή η εντολή δείχνει το είδος του προβλήματος το οποίο είναι πρόβλημα ικανοποίησης χωρίς κάποια συνάρτηση Βελτιστοποίησης. Πρόβλημα ικανοποίησης σημαίνει ότι επιθυμούμε να βρούμε μια τιμή για κάθε μεταβλητή και παράλληλα να ικανοποιούνται όλοι οι περιορισμοί του προβλήματος. Στο συγκεκριμένο παράδειγμα όλες οι μεταβλητές είναι μεταβλητές απόφασης αφού έχουν μπροστά την λέξη *var* άρα, ζητείται από τον επιλυτή Minizinc να αναθέσει μια τιμή σε κάθε μεταβλητή απόφασης καθώς ικανοποιούνται όλοι οι περιορισμοί του προβλήματος.

Η βασική δομή ενός μοντέλου σε γλώσσα Minizinc, αποτελείται από πολλά στοιχεία και το κάθε ένα τελειώνει με ερωτηματικό(;). Συγκεκριμένα, υπάρχουν 6 είδη στοιχείων τα οποία παρουσιάζονται πιο κάτω:

1. Με την λέξη κλειδί *include* επιτρέπεται στο περιεχόμενο ενός άλλου μοντέλου Minizinc να εισαχθεί στο μοντέλο μας. Η λέξη *filename* αναφέρεται στο όνομα του αρχείου το οποίο θα εισαχθεί.

   include *⟨filename⟩*

   Ακόμη, επιτρέπεται σε μεγάλα μοντέλα να χωριστούν σε μικρότερα υπό-μοντέλα καθώς επίσης επιτρέπεται και η ενσωμάτωση των περιορισμών που ορίζονται σε

αρχεία βιβλιοθήκης. Μια εξίσου σημαντική δυνατότητα είναι, το γεγονός ότι στην γλώσσα Minizinc υπάρχει η δυνατότητα χρησιμοποίησης αρχείων με τα οποία μπορούμε να λύσουμε το ίδιο πρόβλημα άλλα με διαφορετικά δεδομένα κάθε φορά. Αυτό επιτυγχάνεται με την εισαγωγή ενός αρχείου με επέκταση ".dzn" την ώρα εκτέλεσης του μοντέλου μας. Ο κώδικας του μοντέλου έχει επέκταση ".mzn".

2. Οι μεταβλητές στην γλώσσα Minizinc χωρίζονται σε δύο κατηγορίες. Στην πρώτη κατηγορία ανήκουν αυτές στις οποίες έχει εκχωρηθεί μια σταθερή τιμή στο μοντέλο ή μέσα από ένα αρχείο δεδομένων τους έχει δοθεί τιμή. Στην δεύτερη κατηγορία ανήκουν οι μεταβλητές απόφασης οι οποίες παίρνουν τιμή όταν το μοντέλο λυθεί.

3. Η εκχώρηση τιμής σε μια μεταβλητή έχει την μορφή:

   ⟨variable⟩ = ⟨expression⟩;

   Τιμές μπορούν να εκχωρηθούν και στις μεταβλητές απόφασης και σε αυτή την περίπτωση η ανάθεση είναι ισοδύναμη με το γράψιμο του περιορισμού

   constraint ⟨variable⟩ = ⟨expression⟩;

4. Η λέξη κλειδί «constraint» είναι το σημείο κλειδί της γλώσσας. Η σύνταξη ενός περιορισμού είναι η εξής: *«constraint <Boolean Expression>;».* Υπάρχουν πολλών ειδών περιορισμοί όπως είναι οι global constraints, η Built in assert κλπ. Οι global περιορισμοί βρίσκονται σε μια βιβλιοθήκη την ***"globals.mzn"*** και κάνοντας ***«include "globals.mzn"*** ;», μπορούμε ανά πάσα στιγμή να χρησιμοποιήσουμε στο μοντέλο μας όποιο περιορισμό ανήκει στους global περιορισμούς. Ένα παράδειγμα αυτού του είδους περιορισμών είναι ο περιορισμός ***« alldiferrent()»*** ο οποίος απαιτεί ότι όλες οι μεταβλητές που εμφανίζονται στις παρενθέσεις του θα έχουν διαφορετικές τιμές.

5. Το σημείο «solve " " » στο μοντέλο, προσδιορίζει τι είδους λύσης αναζητούμε. Υπάρχουν τρία είδη αναζήτησης λύσης. Το πρώτο αναφέρεται στο «***solve** satisfy;*» το οποίο αναφέρεται σε λύση προβλήματος ικανοποίησης. Το δεύτερο είναι το «***solve** minimize φ;*», το οποίο αναφέρεται στην συνάρτηση βελτιστοποίησης *φ*. Συγκεκριμένα, αναζητούμε μια λύση η οποία ελαχιστοποιεί την συνάρτηση *φ* ικανοποιώντας παράλληλα και τους περιορισμούς του προβλήματος και προσδιορίζει ότι το πρόβλημα είναι πρόβλημα ελαχιστοποίησης. Αντίστοιχα, για το «***solve** maximize φ;*», το οποίο αναφέρεται στην συνάρτηση βελτιστοποίησης *φ και*

αναζητούμε μια λύση η οποία μεγιστοποιεί την συνάρτηση *φ* ικανοποιώντας παράλληλα και τους περιορισμούς του προβλήματος και προσδιορίζει ότι το πρόβλημα είναι πρόβλημα μεγιστοποίησης.

6. Για την έξοδο των στοιχείων και για την παρουσίαση των αποτελεσμάτων της εκτέλεσης του μοντέλου, χρησιμοποιείται η πιο κάτω εντολή η οποία έχει τη μορφή:

   output *[⟨string expression⟩, ・ ・ ・ , ⟨string expression⟩];*

Επιπλέον, μια άλλη δυνατότητα που παρέχεται στον προγραμματιστή είναι η χρησιμοποίηση πινάκων και συνόλων. Με την έκφραση *« **array**[1..10] **of var int**: K ;»* ορίζεται ένας μονοδιάστατος πίνακας 10 θέσεων με όνομα K με πεδίο τιμών τους ακεραίους αριθμούς. Με την έκφραση ***«set of int: Products = 1..10;»*** ορίζεται ένα ακέραιο σύνολο με όνομα *Products* μεγέθους 10.

Στην γλώσσα αυτή υπάρχουν πολλών ειδών περιορισμοί. Ο πιο κύριος περιορισμός είναι ο περιορισμός ***«forall».*** Η σύνταξη του περιορισμού αυτού είναι της μορφής, ***«constraint forall(i in Products)(K[i]<15);)».*** Η δήλωση αυτή περιορίζει κάθε θέση (forall) του πίνακα K να έχει τιμή μικρότερη του 15.[7]

6.2.2 Ο επιλυτής Minisat+

Με την πάροδο των τελευταίων ετών, τα εργαλεία επίλυσης(solver) προβλημάτων ικανοποιησιμότητας προτασιακής λογικής έχουν εξελιχτεί σε μεγάλο βαθμό. Λόγω της ραγδαίας εξέλιξης τους και χρήσης τους σε ευρύτερα και περισσότερα πεδία εφαρμογής, παρουσιάστηκε η ανάγκη χρήσης περιορισμών πέραν από την προτασιακή λογική του SAT. Μια δημοφιλής προσέγγιση είναι η επέκταση του εργαλείου επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής για να είναι σε θέση να χειριστεί και άλλα είδη περιορισμών όπως είναι η ψευδοδυαδικοί περιορισμοί, ή ακόμη να εργαστεί σε διάφορα πεδία ορισμών, όπως για παράδειγμα τα πεπερασμένα σύνολα.[9]

Ο επιλυτής Minisat+ είναι ένα εργαλείο επίλυσης προτασιακής ικανοποιησιμότητας που υποστηρίζει ψευδοδυαδικούς περιορισμούς. Είναι μια επέκταση του προτασιακού επιλυτή Minisat. Απώτερος σκοπός του ήταν να μπορεί να υποστηρίζει ψειδοδυαδικούς περιορισμούς όπως για παράδειγμα γραμμικούς περιορισμούς της μορφής 2 X - Y <5.

Ο επιλυτής Minisat+ επιλύει ψευδοδυαδικά προβλημάτων, τα προβλήματα αυτά είναι επίσης γνωστά και ως προβλήματα προγραμματισμού γραμμικών ακεραίων 0-1 περιορισμών (integer linear programming ILP). Έτσι πρέπει να δούμε πώς ένα εργαλείο επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής μπορεί να χρησιμοποιηθεί για την επίλυση Ψευδοδυαδικών προβλημάτων μεταφράζοντας τα σε προτάσεις περιορισμών.

Συγκεκριμένα, ένας Ψευδοδυαδικός περιορισμός αποτελεί μια ανισότητα σε ένα γραμμικό συνδυασμό των δυαδικών μεταβλητών. Η ανισότητα είναι της μορφής $C_0 * p_0 + C_{1*} p_1 + ... + C_{n-1*} p_{n-1} \geq C_n$ όπου, οι μεταβλητές/όροι $p_0,\ldots,p_{n-1}$ ανήκουν στο πεδίο $\{0,1\}$ και C_i ακέραιοι αριθμοί. Ένας όρος παίρνει την τιμή 1 εάν αποτιμάται με την σταθερά true και τιμή 0 εάν αποτιμάται με την σταθερά False. Η αριστερή πλευρά «$C_0 * p_0 + C_1 * p_1 + ... + C_{n-1*pn-1}$» είναι οι όροι και στην δεξιά πλευρά είναι μια η σταθερά C_n. Ο συντελεστής C_i ενεργοποιείται στο πλαίσιο μιας μερικής εκχώρησης εάν στον όρο p_i ανατεθεί η τιμή αληθείας True. Ένας ψευδοδυαδικός περιορισμός ικανοποιείται εάν ικανοποιείται η ανισότητα η οποία έχει την μορφή που προαναφέρθηκε. Συγκεκριμένα, εάν μετά την ανάθεση των τιμών το άθροισμα των συντελεστών υπερβαίνει ή είναι ίσο με την σταθερά C_n που βρίσκεται στην δεξιά πλευρά. Ο χαρακτήρας '-' δηλώνει ότι είναι το συμπλήρωμα (αρνητική εμφάνιση)της μεταβλητής και χαρακτήρας '+' δηλώνει ότι είναι η θετική εμφάνιση της μεταβλητής.[9]

Αν όλες οι σταθερές C_i είναι 1, τότε ο ψευδοδυαδικός περιορισμός είναι ισοδύναμός με μια τυποποιημένη πρόταση ικανοποιησιμότητας προτασιακής λογικής. Οι πιο πάνω μορφή ανισοτήτων αποτελεί την έκφραση των περιορισμών κάποιου προβλήματος έτσι ώστε να είναι αντιληπτοί από τον επιλυτή Minisat+.

Συνεχίζοντας, θα γίνει αναφορά στο πως χωρίς την τροποποίηση της διαδικασίας επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής οι ψευδοδυαδικοί περιορισμοί μπορούν να μεταφραστούν σε προβλήματα ικανοποιησιμότητας προτασιακής λογικής. Εφαρμόστηκαν κάποιες τεχνικές μετάφρασης έτσι ώστε να γίνουν οι περιορισμοί αντιληπτοί από το εργαλείο επίλυσης το ονομαζόμενο Minisat+ το οποίο περιλαμβάνει επίσης υποστήριξη συνάρτηση βελτιστοποίησης που έχει ως αποτέλεσμα την βέλτιστη λύση του προβλήματος.

Πρόσφατα έχουν δημιουργηθεί πολλά εργαλεία επίλυσης ψευδοδυαδικών περιορισμών, τα οποία επεκτείνουν τις τεχνικές επίλυσης προτασιακής ικανοποιησιμότητας (SAT) σε επίλυση ψευδοδυαδικών περιορισμών.

Στο Minisat+ εφαρμόζονται οι τεχνικές κανονικοποίησης κατά τη διάρκεια της μεταγλώττισης των ψευδοδυαδικών περιορισμών όπως αναφέρθηκαν σε προηγούμενο κεφάλαιο στο υποκεφάλαιο 3.7.4. Όταν όλοι οι περιορισμοί έχουν μεταγλωττιστεί με βάση τις τεχνικές κανονικοποίησης προκύπτουν κάποια συμπεράσματα. Όπως για παράδειγμα θέτεται η μεταβλητή x = True, αν όμως η μεταβλητή πάρει την τιμή False αυτομάτως το πρόβλημα θεωρείται μη ικανοποιήσιμο. Αν η τιμή τώρα της μεταβλητής είναι ίση με x= True τότε η συγκεκριμένη μεταβλητή αφαιρείται από όλους τους περιορισμούς. Όπως παρατηρούμε αυτά τα συμπεράσματα είναι τετριμμένα, αφού είναι αυτονόητα.[9]

Τελευταία μετατροπή που εκτελείται είναι μια μετατροπή σε ψευδοδυαδικό επίπεδο έτσι ώστε να μειώσει το μέγεθος των μεταφράσεων σε προβλήματα ικανοποιησιμότητας προτασιακής λογικής.

6.2.3 Ο επιλυτής Bsolo

Ακόμη ένας επιλυτής ο οποίος επιλύει ψειδοδυαδικούς περιορισμούς είναι ο Bsolo. Σε αυτό τον επιλυτή, υλοποιείται ένας εντελώς διαφορετικός αλγόριθμος σε σχέση με τον επιλυτή Minisat+. Ο αλγόριθμος αυτός χρησιμοποιεί διάφορα στοιχεία από τους αλγόριθμους επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής σε μια διαδικασία η όποια υλοποιεί τον αλγόριθμο Κλάδου και Φράγματος(Branch and Bound).[4]

Ο αλγόριθμος που χρησιμοποιεί ο Bsolo ενσωματώνει τα βασικότερα χαρακτηριστικά από τις δύο προσεγγίσεις. Συγκεκριμένα, προσπαθεί να χρησιμοποιεί το χαμηλότερο φράγμα από το αλγόριθμό του Κλάδου και Φράγματος και τις τεχνικές κλαδέματος αναζήτησης από τους αλγόριθμους επίλυσης προβλημάτων ικανοποιησιμότητας προτασιακής λογικής συμπεριλαμβανομένου και της μη χρονολογικής οπισθοδρόμησης στο δένδρο αναζήτησης και τους μηχανισμούς μάθησης στηριζόμενους σε συγκρούσεις.[4]

Η μορφή ανισοτήτων η οποία αποτελεί έκφραση των ανισοτήτων των περιορισμών κάποιου προβλήματος έτσι ώστε να είναι αντιληπτοί από τον επιλυτή Bsolo είναι η μορφή γραμμικών

ψευδοδυαδικών περιορισμών. Συγκεκριμένα, η μορφή είναι η ίδια με την μορφή του επιλυτή Minisat+ με την διαφορά ότι ο επιλυτής αυτός αντιλαμβάνεται το σύμβολο κενό ‘ ‘ αντί του συμβόλου ‘*’. Έτσι η ανισότητα τροποποιείται ως εξής $C_0\, p_0 + C_1 p_1 + \ldots + C_{n-1}\, p_{n-1} \geq C_n$.

Ο αλγόριθμος Κλάδου και Φράγματος αυτός, είναι μια εντελώς διαφορετική προσέγγιση σε προβλήματα βελτιστοποίησης η οποία χρησιμοποιεί ευρετικές μεθόδους που αφορούν την συνάρτηση f με σκοπό να κατευθύνει καλύτερα την αναζήτηση.

Τα κυριότερα βήματα του αλγορίθμου BSOLO είναι τα εξής:

1. Γίνεται αρχικοποίηση του άνω ορίου με την μεγαλύτερη δυνατή τιμή, που ορίζεται από το άθροισμα $ub = \sum_{j=1}^{n} cj+1$
2. Στην συνέχεια με την βοήθεια της συνάρτησης consistent_state γίνεται ο έλεγχος για να αποφασιστεί εάν η παρούσα κατάσταση προκαλεί σύγκρουση. Για να αποφασιστεί κατά πόσο υπάρχει σύγκρουση χρησιμοποιείται η μέθοδος μοναδιαίας μετάδοσης (unit propagation).
3. Στην περίπτωση όπου εντοπιστεί σύγκρουση γίνεται το κάλεσμα μιας βοηθητικής συνάρτησης για ανάλυση της σύγκρουσης.
4. Εάν έχει βρεθεί μια λύση με βάση τους περιορισμούς, τότε γίνεται ενημέρωση του άνω ορίου με βάση το άθροισμα ub=$\sum_{j=1}^{n} cj * xj$
5. Για να μειωθεί η τιμή της συγκεκριμένης λύσης χρησιμοποιείται η μέθοδος της οπισθοδρόμησης με απώτερο σκοπό να εντοπιστεί καλύτερη λύση με χαμηλότερο κόστος.
6. Στο τέλος εκτιμάται το κάτω όριο το οποίο δίνεται στην παρούσα ανάθεση τιμών στις μεταβλητές.
7. Εάν η τιμή αυτή είναι υψηλότερη ή ίση από το τρέχων άνω όριο τότε η διαδικασία ανάλυσης της σύγκρουσης καλείται να προσδιορίσει σε πιο σημείο στο δένδρο αναζήτησης πρέπει να γίνει οπισθοδρόμηση.
8. Στην συνέχεια επιστρέφει στο δεύτερο βήμα του αλγορίθμου.

Με τον πιο πάνω αλγόριθμο είναι δυνατόν να προκύψουν δύο τύποι συγκρούσεων. Ο πρώτος τύπος σύγκρουσης είναι οι λογικές συγκρούσεις οι οποίες ισχύουν όταν τουλάχιστον ένας από τους περιορισμούς του προβλήματος είναι μη ικανοποιήσιμος. Όταν παρουσιαστούν οι λογικές συγκρούσεις ο ψευδοδυαδικός περιορισμός ο οποίος είναι μη ικανοποιήσιμος δίνεται στην διαδικασία ανάλυσης της σύγκρουσης η οποία μαθαίνει μια νέα πρόταση και καθορίζει σε πιο σημείο της διαδικασίας αναζήτησης θα πρέπει να οπισθοδρομήσει. Ο δεύτερος τύπος

σύγκρουσης είναι οι συγκρούσεις φράγματος οι οποίες ισχύουν όταν το κάτω όριο είναι μεγαλύτερο ή ίσο με το άνω όριο.

Επίσης, είναι δυνατή η μάθηση ενός καινούργιου ψευδοδυαδικού περιορισμού. Η μάθηση των προτάσεων μπορεί να αποφύγει την επίσκεψη αχρείαστων μέρων του δένδρου αναζήτησης, αργότερα κατά τη διαδικασία αναζήτησης. Έτσι, για αυτό τον λόγο στα προβλήματα ικανοποιησιμότητας προτασιακής λογικής και προβλήματα βελτίωσης ψευδοδυαδικών προβλημάτων η διαδικασία αυτή είναι συχνά πολύ αποτελεσματική, επιτρέποντας μη χρονολογική οπισθοδρόμηση.

6.3 Περιορισμοί Προβλήματος

Όπως έχουμε αναφέρει και στο κεφάλαιο 5, απώτερος σκοπός του προβλήματος αυτόνομης και εικονικής διαχείρισης των πόρων είναι η αυτόνομη διαχείριση των πόρων για τον έλεγχο του εικονικού περιβάλλοντος έτσι που να αποσυνδέεται η τροφοδότηση(vm provisioning) από την δυναμική τοποθέτηση των εικονικών μηχανών(vm placement). Αυτή η αυτόνομη διαχείριση στοχεύει στην βελτιστοποίηση της συνάρτησης ικανοποίησης(global utility function) όπως επίσης και την δυνατότητα αυτοματοποίησης δυναμικών τροφοδοτήσεων και τοποθετήσεων των εικονικών μηχανών. Η συνάρτηση ικανοποίησης(global utility function) προσδιορίζει την ικανοποίηση της κάθε εφαρμογής με συγκεκριμένη κατανομή πόρων(CPU, RAM) σε σχέση με τους στόχους επίδοσης.

Το πρόβλημα αποτελείται από δύο φάσεις ή καλύτερα δύο επίπεδα, τα δύο αυτά επίπεδα τα οποία πρέπει να διαχειρίζονται είναι η τροφοδότηση εικονικών μηχανών(VM Provisioning) και η τοποθέτηση εικονικών μηχανών(VM Placement). Η φάση τροφοδότησης εικονικών μηχανών είναι υπεύθυνη για την κατανομή της χωρητικότητας των πόρων με την μορφή εικονικές μηχανές σε εφαρμογές και η φάση τοποθέτησης εικονικών μηχανών είναι υπεύθυνη για την αντιστοίχιση των εικονικών μηχανών σε φυσικές μηχανές.

6.3.1 Φάση Τροφοδότησης Εικονικών Μηχανών - Vm Provisioning

Όπως αναφέραμε και πιο πάνω στόχος αυτής της φάσης είναι η εύρεση του διανύσματος κατανομής εικονικών μηχανών, N_i για κάθε εφαρμογή μεγιστοποιώντας την συνάρτηση ικανοποίησης(global utility value -UGlobal).

Παράμετροι του προβλήματος

- $A=(\alpha_1,\alpha_2,\alpha_i,..,\alpha_m)$ δηλώνει το σύνολο των εφαρμογών.
- $P=(p_1,p_2,p_j,..,p_q)$ δηλώνει το σύνολο των φυσικών μηχανών.
- Κάθε κατηγορία εικονικών μηχανών έχει συγκεκριμένη ικανότητα σε Cpu και μνήμη Ram. Υπάρχουν c κατηγορίες εικονικών μηχανών που διατίθενται μεταξύ του διανύσματος $S=(S_1,S_2,S_k..,S_c)$, όπου $S_k=(S_k^{Cpu}, S_k^{Ram})$, S_k^{Cpu} προσδιορίζει την ικανότητα της Cpu και S_k^{Ram} προσδιορίζει την ικανότητα μνήμης (Ram) της εικονική μηχανής k.
- Τα διανύσματα C_j^{cpu} και C_j^{ram} προσδιορίζουν την ικανότητα της φυσικής μηχανής p_j σε Cpu και σε Ram αντίστοιχα.
- Άνω φράγμα για τον αριθμό των εικονικών μηχανών σε κάθε κατηγορία $N_i^{max}=(n_{i1}^{max},n_{i2}^{max},n_{ik}^{max},n_{im}^{max})$.
- Το N_i είναι το διάνυσμα κατανομής εικονικών μηχανών της εφαρμογής α_i όπου, $N_i=(n_{i1},n_{i2},\ldots,n_{ik},n_{im})$ - n_{ik} είναι ο αριθμός των εικονικών μηχανών της κατηγορίας S_k που αποδίδεται στην εφαρμογή α_i.
- Το επίπεδο των πόρων (recource level function), U_i για την εφαρμογή α_i ορίζεται ως $U_i=f_i(N_i)= c1_{i1}*n_{i1} + c1_{i2}*n_{i2} + ... + c1_{im}*n_{im}$.
- Το λειτουργικό κόστος για την εφαρμογή α_i ορίζεται ως $Cost_i=c2_{i1}*n_{i1} + c2_{i2}*n_{i2} + ... + c2_{im}*n_{im}$.
- Ο συνολικός αριθμός εικονικών μηχανών για κάθε εφαρμογή εκφράζεται από το διάνυσμα $T_i^{max.}$.

Στην παρούσα διατύπωση του προβλήματος υποθέτουμε ότι οι εφαρμογές έχουν σταθερό φορτίο, υπό την έννοια ότι οι απαιτήσεις αυτές δεν αυξομειώνονται. Συγκεκριμένα, οι απαιτήσεις των εφαρμογών σε πόρους παραμένουν σταθερές κατά την διάρκεια εκτέλεσης τους.

Οι περιορισμοί για την φάσης τροφοδότησης εικονικών μηχανών περιγράφονται λεπτομερώς πιο κάτω.

Περιορισμός 1

Ο πρώτος περιορισμός εκφράζεται από την ανίσωση $nik \leq nikmax$, όπου το i αναφέρεται στις εφαρμογές και το k αναφέρεται στις κατηγορίες εικονικών μηχανών. Με αυτό τον περιορισμό εξασφαλίζουμε ότι η ανάθεση η οποία θα γίνει στο διάνυσμα N_i για

κάθε εφαρμογή και για κάθε κατηγορία εικονικών μηχανών, δεν θα ξεπερνά το άνω φράγμα για τον αριθμό των εικονικών μηχανών σε κάθε κατηγορία.

Περιορισμός 2

Ο δεύτερος περιορισμός εκφράζεται από την ανίσωση $\sum_{k=1}^{c} ni_{\mathrm{k}} \leq \mathrm{T_i}^{\max}$, όπου το i αναφέρεται στις εφαρμογές και το k αναφέρεται στις κατηγορίες εικονικών μηχανών. Με αυτόν το περιορισμό εξασφαλίζουμε ότι το συνολικό άθροισμα των εικονικών μηχανών για κάθε εφαρμογή θα είναι μικρότερο ή ίσο του συνολικού αριθμού των εικονικών μηχανών που απαιτεί η συγκεκριμένη εφαρμογή.

Περιορισμός 3

Ο περιορισμός αυτός, δεν αναφέρεται στην θεωρία του προβλήματος άλλα προστέθηκε κατά την υλοποίηση, για την καλύτερη κατανομή των εικονικών μηχανών. Ο περιορισμός αυτός αναφέρει ότι σε κάθε εφαρμογή θα πρέπει να ανατεθεί τουλάχιστον μια εικονική μηχανή. Έτσι ο περιορισμός αυτός εκφράζεται από την ανίσωση $\sum_{k=1}^{c} n_{\mathrm{ik}} \geq 1$.

Περιορισμοί 4 και 5

Οι δύο πιο κάτω περιορισμοί, είναι παρόμοιοι αλλά ο ένας αναφέρεται στην ικανότητα σε Cpu και ο άλλος στην μνήμη Ram. Οι δύο αυτοί περιορισμοί εκφράζονται από τις ακόλουθες ανισώσεις $\sum_{i=1}^{m}\sum_{k=1}^{c} \mathrm{n_{ik}}*\mathrm{S_k}^{\mathrm{cpu}} \leq \sum_{\mathrm{j}=1}^{q} \mathrm{C_j}^{\mathrm{cpu}}$ και $\sum_{i=1}^{m}\sum_{k=1}^{c} \mathrm{n_{ik}}*\mathrm{S_k}^{\mathrm{ram}} \leq \sum_{\mathrm{j}=1}^{q} \mathrm{C_j}^{\mathrm{ram}}$. Αυτοί οι περιορισμοί αναφέρουν ότι το συνολικό άθροισμα για κάθε κατηγορία εικονικής μηχανής για την κατανομή που θα γίνει στο διάνυσμα $\mathrm{N_i}$ επί την ικανότητα της Cpu (ή Ram αντίστοιχα) της κατηγορίας αυτής εικονικών μηχανών, δεν πρέπει να ξεπερνά το συνολικών άθροισμα των ικανοτήτων των φυσικών μηχανών σε Cpu (ή Ram αντίστοιχα).

Περιορισμοί 6 και 7

Το επίπεδο των πόρων(recource level function) $\mathrm{U_i}$ για την εφαρμογή α_i ορίζεται ως $\mathrm{U_i}=\mathrm{f_i}(\mathrm{N_i})= \mathrm{c1_{i1}}*\mathrm{n_{i1}} + \mathrm{c1_{i2}}*\mathrm{n_{i2}} + ... + \mathrm{c1_{im}}*\mathrm{n_{im}}$. Η εξίσωση αυτή εκφράζει το συνολικό επίπεδο των πόρων που θα έχουν οι εφαρμογές ανάλογα με τον συντελεστή/βαρύτητα($\mathrm{c1_{im}}$) που θα αποδοθεί σε κάθε εφαρμογή για κάθε κατηγορία εικονικών μηχανών. Το λειτουργικό κόστος για την εφαρμογή α_i ορίζεται ως $\mathrm{Cost_i}=\mathrm{c2_{i1}}*\mathrm{n_{i1}} + \mathrm{c2_{i2}}*\mathrm{n_{i2}} + ... + \mathrm{c2_{im}}*\mathrm{n_{im}}$. Η εξίσωση αυτή αντιπροσωπεύει το συνολικό λειτουργικό κόστος που θα έχουν οι εφαρμογές ανάλογα με τον συντελεστή/βαρύτητα($\mathrm{c2_{im}}$) που θα αποδοθεί σε κάθε εφαρμογήγια κάθε κατηγορία εικονικών μηχανών.

Συνάρτηση Βελτιστοποίησης - Ικανοποίηση των εφαρμογών (Global Utility Function)

Όπως έχουμε αναφέρει και προηγουμένως σκοπός αυτής της φάσης είναι να γίνει μια κατανομή στο διάνυσμα N_i για κάθε εφαρμογή και να προσδιοριστεί πόσες εικονικές μηχανές από κάθε κατηγορία χρειάζεται η κάθε εφαρμογή. Αυτή η κατανομή θα πρέπει να γίνει με τέτοιο τρόπο έτσι ώστε να μεγιστοποιείται η συνάρτηση ικανοποίησης των εφαρμογών(Global Utility Function) η οποία δίνει ένα μέτρο ικανοποίησης της εφαρμογής με συγκεκριμένη κατανομή πόρων(Cpu,Ram). Η συνάρτηση αυτή εκφράζεται από την πιο κάτω εξίσωση:

$$\text{Uglobal= maximize } (10\text{-e}) * \sum_{i=1}^{m}(a_i * U_i) - (e * \text{total}) * \sum_{i=1}^{m}(Cost_i)$$

όπου a_i είναι η βαρύτητα κάθε εφαρμογής, total είναι το άθροισμα όλων των βαρυτήτων των εφαρμογών και e συμβολίζει τον συντελεστής κόστους. Σκοπός της πιο πάνω εξίσωσης είναι να ισορροπηθεί η διαφορά ανάμεσα στο επίπεδο των πόρων και στο λειτουργικό κόστος το οποίο θα έχει η κάθε εφαρμογή σε κάθε κατηγορία εικονικών μηχανών.

6.3.2 Φάση Τοποθέτησης Εικονικών Μηχανών - Vm Placement

Είσοδος σε αυτή την φάση είναι το διάνυσμα N_i το οποίο προκύπτει από την φάση τροφοδότησης εικονικών μηχανών. Το διάνυσμα N_i, προσδιορίζει ποιος είναι ο συνολικός αριθμός των εικονικών μηχανών που απαιτούνται, αφού όπως έχουμε αναφέρει και πιο πάνω προσδιορίζει πόσες εικονικές μηχανές από κάθε κατηγορία χρειάζεται η κάθε εφαρμογή. Όπως έχει προαναφερθεί στόχος της φάσης αυτής είναι η διάθεση των εικονικών μηχανών σε φυσικές μηχανές προκειμένου να ελαχιστοποιηθεί ο αριθμός των ενεργών φυσικών μηχανών(φάση τοποθέτησης εικονικών μηχανών).

Για την φάση τοποθέτησης εικονικών μηχανών θα ασχοληθούμε με δύο τρόπους υλοποίησης της. Η διάφορα των δύο τρόπων έγκειται στην είσοδο που παίρνουν από την πρώτη φάση τροφοδότησης των εικονικών μηχανών. Ο πρώτος τρόπος υλοποίησης παίρνει ως είσοδο από την φάση τροφοδότησης εικονικών μηχανών έναν αριθμό ο οποίος αντιστοιχεί στο συνολικό άθροισμα ολόκληρου του διανύσματος N_i, ο αριθμός αυτός προσδιορίζει ποιος είναι ο συνολικός αριθμός εικονικών μηχανών που απαιτούνται. Αντίθετα, ο δεύτερος τρόπος υλοποίησης παίρνει ως είσοδο ένα διάνυσμα το οποίο αναφέρεται σε κάθε κατηγορία εικονικής μηχανής. Συγκεκριμένα, παίρνει ένα αριθμό για κάθε κατηγορία εικονικών μηχανών δηλαδή, πόσες εικονικές μηχανές απαιτούνται για κάθε κατηγορία. Το αποτέλεσμα και ο σκοπός των δύο τρόπων υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές ταυτίζονται. Ο λόγος για τον οποίο η φάση τοποθέτησης εικονικών

μηχανών υλοποιήθηκε με δύο τρόπους είναι για να γίνει σύγκριση των δύο τρόπων υλοποίησης στο επόμενο κεφάλαιο της πειραματικής αξιολόγησης και των συμπερασμάτων. Η θεωρία γύρω από τους περιορισμούς αυτής της φάσης είναι η ίδια και στους δύο τρόπους υλοποίησης, το μόνο που αλλάζει είναι ο τρόπος απεικόνισης τους στο κάθε τρόπο επίλυσης του προβλήματος. Πιο κάτω θα παρουσιαστούν οι περιορισμοί με τις μικρό-αλλάγες τους για κάθε τρόπο επίλυσης του προβλήματος.

Ο πρώτος τρόπος υλοποίησης παίρνει ως είσοδο από την φάση τροφοδότησης εικονικών μηχανών έναν αριθμό ο οποίος απεικονίζει το συνολικό άθροισμα ολόκληρου του διανύσματος N_i, ο αριθμός αυτός προσδιορίζει ποιος είναι ο συνολικός αριθμός εικονικών μηχανών που απαιτούνται. Έστω ότι αθροίζοντας το διάνυσμα N_i προκύπτει ο αριθμός u', ο οποίος προσδιορίζει ποιος είναι ο συνολικός αριθμός εικονικών μηχανών.

Ο δεύτερος τρόπος υλοποίησης παίρνει ως είσοδο από την φάση τροφοδότησης εικονικών μηχανών ένα διάνυσμα το οποίο αναφέρεται σε κάθε κατηγορία εικονικής μηχανής. Συγκεκριμένα, παίρνει ένα αριθμό για κάθε κατηγορία εικονικών μηχανών, δηλαδή πόσες εικονικές μηχανές απαιτούνται για κάθε κατηγορία. Έστω το διάνυσμα αυτό είναι το allocate$\{u_1,u_2\ldots,u_c,\}$ όπου u_i είναι ο αριθμός εικονικών μηχανών για την κατηγορία i.

Για τον πρώτο τρόπο υλοποίησης χρησιμοποιήθηκε το διάνυσμα $H_j=(h_{j1},h_{j2},h_{jl},\ldots,h_{ju'})$. Το διάνυσμα αυτό συμβολίζει το σύνολο των εικονικών μηχανών που ανατίθενται στην φυσική μηχανή p_j. Τυπικά η τοποθέτηση εικονικών μηχανών σε φυσικές μηχανές συμβολίζεται ως $H_{jl}=1$ εάν η φυσική μηχανή p_j φιλοξενεί την εικονική μηχανή vm_l. Το H_j είναι δυαδικό διάνυσμα (0 ή 1 τιμές).

Για τον δεύτερο τρόπο υλοποίησης το διάνυσμα $D_j=(d_{j1},d_{j2},d_{j3},\ldots,d_{jc})$, συμβολίζει το σύνολο των εικονικών μηχανών που ανατίθενται στην φυσική μηχανή p_j. Τυπικά, η τοποθέτηση εικονικών μηχανών σε φυσικές μηχανές συμβολίζεται ως, $D_{jl}>1$ εάν η φυσική μηχανή p_j φιλοξενεί έστω και μια εικονική μηχανή τύπου l. Προφανώς το D_j δεν είναι δυαδικό διάνυσμα με τιμές 0 ή 1, αλλά παίρνει τιμές από 0 μέχρι u', τον συνολικό αριθμό εικονικών μηχανών που απαιτούνται(το άθροισμα του διανύσματος N_i).

Το διάνυσμα $R=(r_1,r_2,r_l,..,r_u)$ προσδιορίζει την ικανότητα των πόρων(cpu και ram) όλων των εικονικών μηχανών. Συγκεκριμένα συμβολίζεται με $r_l=(r_l^{cpu},r_l^{ram})$.

Οι περιορισμοί για την φάση τοποθέτησης εικονικών μηχανών περιγράφονται λεπτομερώς πιο κάτω.

Περιορισμός 1 και 2

Οι δύο πιο κάτω περιορισμοί είναι παρόμοιοι, ο ένας αναφέρεται στην Cpu και ο άλλος στην μνήμη Ram. Οι δύο αυτοί περιορισμοί εκφράζονται από τις ακόλουθες ανισώσεις $\sum_{l=1}^{u} r_l^{cpu} * h_{jl} \leq C_j^{cpu}$ *και* $\sum_{l=1}^{u} r_l^{ram} * h_{jl} \leq C_j^{ram}$. Ο δείκτης j αναφέρεται στις φυσικές μηχανές και πρέπει να ισχύει $1 \leq j < q$. Οι δύο αυτοί περιορισμοί αναφέρουν ότι το συνολικό άθροισμα των Cpu των εικονικών μηχανών(αντίστοιχα για Ram), οι οποίες θα ανατεθούν σε μια(στην ίδια) φυσική μηχανή δεν θα πρέπει να ξεπερνά την Cpu (αντίστοιχα για Ram) αυτής της εικονικής μηχανής. Οι δύο πιο πάνω ανισώσεις αναφέρονται στον πρώτο τρόπο υλοποίησης. Για τον δεύτερο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών προκύπτουν οι ανισώσεις $\sum_{l=1}^{c} r_l^{cpu} * d_{jl} \leq C_j^{cpu}$ *και* $\sum_{l=1}^{c} r_l^{ram} * d_{jl} \leq C_j^{ram}$(c είναι αριθμός που προσδιορίζει πόσες κατηγορίες εικονικών μηχανών υπάρχουν).

Περιορισμός 3

Ο τρίτος περιορισμός αποσκοπεί στην καλύτερη κατανομή των εικονικών μηχανών. Ο περιορισμός αυτός αναφέρεται στο ότι κάθε εικονική μηχανή θα πρέπει να ανατεθεί σίγουρα σε κάποια φυσική μηχανή. Έτσι ο περιορισμός αυτός αναφέρεται στην ανίσωση $\sum_{j=1}^{q} h_{jl} = 1$. Η ανίσωση αυτή αναφέρεται στον πρώτο τρόπο υλοποίησης. Για τον δεύτερο τρόπο υλοποίησης προκύπτει η ανίσωση $\sum_{j=1}^{q} d_{jl}$ = allocate[l], η οποία εκφράζει ότι όλες οι εικονικές μηχανές για την κατηγορία l, θα ανατεθούν σε κάποια φυσική μηχανή(όχι απαραίτητα στην ίδια).

Συνάρτηση Βελτιστοποίησης

Όπως έχουμε αναφέρει και προηγουμένως σκοπός αυτής της φάσης είναι να πραγματοποιηθεί η διάθεση των εικονικών μηχανών σε φυσικές μηχανές προκειμένου να ελαχιστοποιηθεί ο αριθμός των ενεργών φυσικών μηχανών. Το διάνυσμα U αναφέρει κατά πόσο μια φυσική μηχανή είναι ενεργή ή όχι δηλαδή, αν τις έχουν ανατεθεί εικονικές μηχανές ή όχι. Έτσι σκοπός είναι να ελαχιστοποιηθεί το άθροισμα $\sum_{j=1}^{q} U_j$.

6.4 Υλοποίηση Περιορισμών στους Επιλυτές

6.4.1 Δεδομένα Εισόδου από τον Χρήστη

Στο πρώτο στάδιο της υλοποίησης του πιο πάνω προβλήματος, ο χρήστης δίνει κάποια δεδομένα εισόδου και μέσα από αυτά παράγονται τυχαίες τιμές για τα διανύσματα έτσι ώστε, μετά μέσα από τους επιλυτές που περιγράφτηκαν πιο πάνω να υλοποιηθούν οι περιορισμοί της κάθε φάσης.

Για την αναπαράσταση των διανυσμάτων χρησιμοποιείται η δομή των πινάκων. Η εισαγωγή των δεδομένων εισόδου και η παραγωγή των τυχαίων τιμών στους πίνακες γίνονται μέσα από την γλώσσα προγραμματίσου C. Συγκεκριμένα, ζητείται από τον χρήστη να δώσει τον αριθμό των εφαρμογών(m), τον αριθμό των κατηγοριών των εικονικών μηχανών(c), τον αριθμό των φυσικών μηχανών(q), το συντελεστή κόστους(e) και την βαρύτητα για κάθε εφαρμογή(a_i). Έτσι μέσα από το πρώτο πρόγραμμα στη C (both_phase_A), ζητείται από τον χρήστη να δώσει τα πιο πάνω δεδομένα και επίσης δημιουργούνται τυχαίες τιμές για τους πίνακες Nmax, Tmax, Ccpu, Cram, S, c1_sintelestis, c2_sintelestis. Στους πίνακες Nmax, Tmax, c1_sintelestis και c2_sintelestis δημιουργούνται τιμές από το 1 μέχρι το 10. Οι πίνακες Nmax, c1_sintelestis και c2_sintelestis είναι δυσδιάστατοι με γραμμές όσες ο αριθμός των εφαρμογών και στήλες όσες ο αριθμός των κατηγοριών εικονικών μηχανών. Ο πίνακας Tmax είναι μονοδιάστατος με μέγεθος όσο ο αριθμός των εφαρμογών. Για τον πίνακα S δημιουργούνται τιμές από το 10 μέχρι το 30. Ο πίνακας S είναι δυσδιάστατος με γραμμές όσες ο αριθμός των κατηγοριών εικονικών μηχανών και δύο στήλες, η πρώτη στήλη αναφέρεται στην ικανότητα Cpu και η δεύτερη στην ικανότητα Ram. Για τους πίνακες Ccpu και Cram δημιουργούνται τιμές από το 30 μέχρι το 100 και είναι μονοδιάστατοι μεγέθους όσο ο αριθμός των φυσικών μηχανών. Έτσι, μέσα από το πρώτο πρόγραμμα στη C (both_phase_A), δημιουργείται το αρχείο εισόδου για τον επιλυτή Minizinc (A_inputA1.dzn), το οποίο αφορά τα δεδομένα για την φάση τροφοδότησης εικονικών μηχανών(vm provisioning phase).

6.4.2 Υλοποίηση στον επιλυτή Minizinc

Όπως έχει περιγραφεί στο πιο πάνω υποκεφάλαιο, μέσα από το πρώτο πρόγραμμα στη C (both_phase_A.c) έχει δημιουργηθεί ένα αρχείο με όνομα A_inputA1.dzn το όποιο αφορά την φάση τροφοδότησης του προβλήματος. Οι περιορισμοί 1-5 και η συνάρτηση βελτιστοποίησης της πρώτης φάσης έχουν υλοποιηθεί στην γλώσσα Minizinc και ο κώδικας για αυτό το κομμάτι της υλοποίησης βρίσκεται στο αρχείο 1omeros.mzn.

Στην υλοποίηση της φάσης τροφοδότησης εικονικών μηχανών(vm provisioning) έχουμε ως μεταβλητή απόφασης τον πίνακα N_i αφού ζητούμενο αυτής της φάσης είναι η κατανομή αυτού του πίνακα δηλαδή, κατανομή για το πόσες εικονικές μηχανές από κάθε κατηγορία εικονικών μηχανών απαιτεί η κάθε εφαρμογή. Ο πίνακας αυτός είναι δυσδιάστατος με γραμμές όσες και ο αριθμός των εφαρμογών(m) και στήλες όσες ο αριθμός των κατηγοριών εικονικών μηχανών(c). Επίσης, έχει πεδίο τιμών τιμές από 0 μέχρι 1000 αλλά στην πραγματικότητα θα παίρνει τιμές από 0..10, αφού υπάρχει ο περιορισμός $n_{ik} \leq n_{ik}^{max}$ και ο πίνακας Nikmax παίρνει τιμές από 1 μέχρι 10. Ο κώδικας σε γλώσσα Minizinc για τον ορισμό της μεταβλητής N_i ως μεταβλητή απόφασης είναι ο εξής:

array[num_efarmoges,num_katigories] of var 0..1000: N;

Όπως προαναφέρθηκε για τις μεταβλητές Nmax, Tmax, Ccpu, Cram, S, c1_sintelestis και c2_sintelestis δίνονται τυχαίες τιμές οι οποίες αποθηκεύονται σε ένα αρχείο με επέκταση «.dzn». Το αρχείο αυτό δίνεται ως δεδομένο εισόδου στο μοντέλο της φάση τροφοδότησης («.mzn αρχείο») και έχει την πιο κάτω μορφή. Το συγκεκριμένο παράδειγμα αναφέρεται σε 5 εφαρμογές και 1 κατηγορία εικονικών μηχανών.

```
m=5;
c=1;
q=20;
Nmax=[|9,
    |3,
    |3,
    |2,
    |5|];
Tmax=[10,8,3,3,5];
S=[|21,26|];
Ccpu=[93,49,36,31,38,74,85,32,70,98,78,89,53,41,96,37,43,43,70,50];
Cram=[44,99,44,88,55,70,88,94,34,41,70,37,55,42,87,31,56,50,56,78];
varitita=[1,2,5,3,7];
c1_sintelestis=[|3,
            |4,
            |10,
            |4,
            |3|];
c2_sintelestis=[|5,
            |3,
            |10,
            |1,
            |8|];
sintelestis_kostous=4;
```

Μετατροπή Περιορισμών της φάσης τροφοδότησης εικονικών μηχανών στη γλώσσα Minizinc

Ο δείκτης i αναφέρεται στις εφαρμογές(m), ο c αναφέρεται στις κατηγορίες εικονικών μηχανών και ο q αναφέρεται στον αριθμό φυσικών μηχανών.

1. $n_{ik} \leq n_{ik}^{max}$

   constraint forall(i in num_efarmoges,j in num_katigories)(N[i,j]<=Nmax[i,j]);

2. $\sum_{k=1}^{c} n_{ik} \leq T_i^{max}$

   constraint forall(i in num_efarmoges)(sum(j in num_katigories) (N[i,j]) <= Tmax[i]);

3. Σε κάθε εφαρμογή θα πρέπει να ανατεθεί τουλάχιστον μια εικονική μηχανή, $\sum_{k=1}^{c} n_{ik} \geq 1$.

   constraint forall(i in num_efarmoges)(exists(j in num_katigories)(N[i,j]!=0));

4. $\sum_{i=1}^{m} \sum_{k=1}^{c} n_{ik} * S_k^{cpu} \leq \sum_{j=1}^{q} C_j^{cpu}$

   *constraint (sum (i in num_efarmoges)(sum(j in num_katigories) (N[i,j]*S[j,1])) <= sum(w in num_pms)(Ccpu[w]));*

5. $\sum_{i=1}^{m} \sum_{k=1}^{c} n_{ik} * S_k^{ram} \leq \sum_{j=1}^{q} C_j^{ram}$

   *constraint (sum (i in num_efarmoges)(sum(j in num_katigories) (N[i,j]*S[j,2])) <= sum(w in num_pms)(Cram[w]));*

6. Το επίπεδο των πόρων(recource level function) U_i για την εφαρμογή α_i ορίζεται ως $U_i=f_i(N_i)= c1_{i1}*n_{i1} + c1_{i2}*n_{i2} + ... + c1_{im}*n_{im}$.

   *constraint forall(i in num_efarmoges) (u[i]=(sum(j in num_katigories)(c1_sintelestis[i,j]*N[i,j])));*

7. Το λειτουργικό κόστος για την εφαρμογή α_i ορίζεται ως $Cost_i=c2_{i1}*n_{i1} + c2_{i2}*n_{i2} + ... + c2_{im}*n_{im}$.

   *constraint forall(i in num_efarmoges) (cost[i]=(sum(j in num_katigories)(c2_sintelestis[i,j]*N[i,j])));*

8. Συνάρτηση Βελτιστοποίησης

Uglobal= maximize (10-e) * $\sum_{i=1}^{m}(a_i*U_{i)}$ - (e*total) * $\sum_{i=1}^{m}(Cost_i)$

*constraint x =(((sintelestis_kostous*total)*sum(i in num_efarmoges)(cost[i]))- ((10 sintelestis_kostous)*sum(i in num_efarmoges)(varitita[i]*u[i])));*
solve minimize x;

Όπως έχει αναφερθεί και προηγουμένως η φάση τροφοδότησης εικονικών μηχανών (vm provisioning) έχει ως έξοδο το διάνυσμα κατανομής N_i το οποίο προσδιορίζει πόσες εικονικές μηχανές από κάθε κατηγορία εικονικών μηχανών απαιτούνται για κάθε εφαρμογή. Έτσι, το αποτέλεσμα εξόδου της φάσης τροφοδότησης εικονικών μηχανών (vm provisioning) από τον επιλυτή Minizinc αποθηκεύεται σε ένα αρχείο(in.txt) έτσι ώστε η έξοδος της φάσης τροφοδότησης εικονικών μηχανών να αποτελεί είσοδο στην φάση τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές(vm placement).

Έτσι, πάλι μέσα από την γλώσσα προγραμματισμού C διαβάζεται το αποτέλεσμα (από το αρχείο in.txt)της φάσης τροφοδότησης εικονικών μηχανών(vm provisioning) και κωδικοποιείται ανάλογα με σκοπό να εισαχθεί στην φάση τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές(vm placement).Συγκεκριμένα, μέσα από το πρόγραμμα both_phase_B.c διαβάζεται το διάνυσμα κατανομής N_i που αποφασίζει ο επιλυτής Minizinc για την φάση τροφοδότησης και κωδικοποιείται ανάλογα για την δεύτερη φάση. Όπως προαναφέραμε η φάση τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές(vm placement) είναι υλοποιημένη με δύο τρόπους. Ο πρώτος τρόπος παίρνει ως είσοδο από την πρώτη φάση ένα αριθμό ο οποίος απεικονίζει το συνολικό άθροισμα ολόκληρου του διανύσματος N_i, ο αριθμός αυτός προσδιορίζει ποιος είναι ο συνολικός αριθμός εικονικών μηχανών που απαιτούνται. Έστω ότι αθροίζοντας το διάνυσμα N_i προκύπτει ο αριθμός u', ο οποίος προσδιορίζει ποιος είναι ο συνολικός αριθμός εικονικών μηχανών. Ο δεύτερος τρόπος παίρνει ως είσοδο από την φάση τροφοδότησης ένα διάνυσμα το οποίο αναφέρεται σε κάθε κατηγορία εικονικής μηχανής. Συγκεκριμένα, παίρνει ένα αριθμό για κάθε κατηγορία εικονικών μηχανών, δηλαδή πόσες εικονικές μηχανές απαιτούνται για κάθε κατηγορία. Έστω το διάνυσμα αυτό είναι το allocate$\{u_1,u_2\ldots,u_c\}$, όπου u_i είναι ο αριθμός εικονικών μηχανών για την κατηγορία i και τοποθετείται σε ένα πίνακα.

Όσων αφορά το διάνυσμα R=$(r_1,r_2,r_l,..,r_u)$ το οποίο προσδιορίζει την ικανότητα των πόρων(Cpu και Ram) όλων των εικονικών μηχανών, συσχετίζεται με τον πίνακα S και την

κατανομή που γίνεται στην φάση τροφοδότησης εικονικών μηχανών(vm provisioning). Ο πίνακας S απεικονίζει την ικανότητα σε Cpu και Ram της κάθε κατηγορίας εικονικών μηχανών. Έστω ότι η κατανομή για την φάση τροφοδότησης εικονικών μηχανών η οποία έγινε για κάθε κατηγορία εικονικών μηχανών είναι της μορφής allocate[u_1,u_2,…,u_c], όπου u_i είναι ο αριθμός εικονικών μηχανών για την κατηγορία i και τοποθετείται σε ένα πίνακα. Έτσι, για τον πρώτο τρόπο υλοποίησης της φάση τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές ο πίνακας R έχει την εξής μορφή: θα είναι μεγέθους $u_1+u_2+..+u_c$ και θα υπάρχουν u_1 θέσεις στον πίνακα με τιμή S_1, u_2 θέσεις με τιμή S_2 , …, u_c θέσεις με τιμή S_c. Για τον δεύτερο τρόπο υλοποίησης της φάση τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές ο πίνακας R θα είναι μεγέθους όσο και το c(ο αριθμός των κατηγοριών εικονικών μηχανών) και η θέση 1 η οποία αφορά την πρώτη κατηγορία θα έχει τιμή S_1, η θέση 2 οποία αφορά την δεύτερη κατηγορία θα έχει τιμή S_2, ….,η θέση c οποία αφορά την τελευταία κατηγορία και θα έχει τιμή S_c.

Έτσι, μέσα από το πρόγραμμα στην C, both_phase_B δημιουργούνται δύο αρχεία εισόδου για τους δύο τρόπους υλοποίησης της φάση τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές, το αρχείο B_inputB1.dzn και το αρχείο B_inputB1_btropos.dzn. Οι περιορισμοί 1-3 και η συνάρτηση βελτιστοποίησης της φάσης τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές έχουν υλοποιηθεί στην γλώσσα Minizinc. Οι δύο τρόποι υλοποίησης για το Minizinc είναι υλοποιημένοι στα αρχεία 2o_meros_B.mzn και 2_btrospos.mzn.

Τώρα ας δούμε μέσα από ένα παράδειγμα την μορφή την οποία θα έχουν τα αρχεία για τους δύο τρόπους υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές. Όπως έχει αναφερθεί, ο πρώτος τρόπος παίρνει ως είσοδο από την φάση τροφοδότησης εικονικών μηχανών ένα αριθμό ο οποίος προσδιορίζει το συνολικό άθροισμα ολόκληρου του διανύσματος N_i, ο αριθμός αυτός προσδιορίζει ποιος είναι ο συνολικός αριθμός εικονικών μηχανών που απαιτούνται. Έστω ότι αθροίζοντας το διάνυσμα N_i προκύπτει ο αριθμός u’, ο οποίος προσδιορίζει ποιος είναι ο συνολικός αριθμός εικονικών μηχανών. Έστω ότι η έξοδος της φάσης τροφοδότησης εικονικών μηχανών για την κατανομή στον πίνακα N_i είναι η εξής 1 ,3 ,1 ,2 , 1.

Έτσι ο αριθμός u’ έχει τιμή ίση με 8(u’=8). Έτσι λοιπόν, το αρχείο εισόδου για την υλοποίηση με τον πρώτο τρόπο της φάσης τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές έχει την πιο κάτω μορφή:

```
q=20; //Αριθμός φυσικών μηχανών
Ccpu=[93,49,36,31,38,74,85,32,70,98,78,89,53,41,96,37,43,43,70,50];
Cram=[44,99,44,88,55,70,88,94,34,41,70,37,55,42,87,31,56,50,56,78];
u_vms=8; // Ο αριθμός u΄-συνολικός αριθμός εικονικών μηχανών που απαιτούνται
R=[|21,26,
  |21,26,
  |21,26,
  |21,26,
  |21,26,
  |21,26,
  |21,26|];
```

Όπως παρατηρούμε από το πιο πάνω παράδειγμα αρχείου, ο πίνακας R είναι μεγέθους 8 με τιμές για όλες τι θέσεις του 21,26 αφού μόνο ένας τύπος εικονικής μηχανής υπάρχει.

Ο δεύτερος τρόπος υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές παίρνει ως είσοδο από την φάση τροφοδότησης εικονικών μηχανών ένα διάνυσμα το οποίο αναφέρεται σε κάθε κατηγορία εικονικής μηχανής. Συγκεκριμένα, παίρνει ένα αριθμό για κάθε κατηγορία εικονικών μηχανών, δηλαδή πόσες εικονικές μηχανές απαιτούνται για κάθε κατηγορία. Έστω το διάνυσμα αυτό είναι το allocate$\{u_1,u_2\ldots,u_c\}$, όπου u_i είναι ο αριθμός εικονικών μηχανών για την κατηγορία i και τοποθετείται σε ένα πίνακα. Έτσι με βάση το παράδειγμα που έχει δοθεί πιο πάνω, ότι δηλαδή η έξοδος της φάσης τροφοδότησης εικονικών μηχανών για την κατανομή στον πίνακα N_i είναι 1, 3, 1, 2, 1, τότε έχουμε ότι το διάνυσμα allocate θα είναι μιας θέσης αφού μόνο ένας τύπος εικονικών μηχανών υπάρχει. Έτσι λοιπόν, το αρχείο εισόδου για την υλοποίηση με τον δεύτερο τρόπο της φάση τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές (vm placement) έχει την εξής μορφή :

```
q=20; // Αριθμός φυσικών μηχανών
Ccpu=[93,49,36,31,38,74,85,32,70,98,78,89,53,41,96,37,43,43,70,50];
Cram=[44,99,44,88,55,70,88,94,34,41,70,37,55,42,87,31,56,50,56,78];
u_vms=8; /Συνολικός αριθμός εικονικών μηχανών που απαιτούνται
type_vms=1; // αριθμός κατηγοριών εικονικών μηχανών
allo=[8]; // διάνυσμα allocate
R=[|21,26,|];
```

Όπως παρατηρούμε από το πιο πάνω παράδειγμα αρχείου, ο πίνακας R είναι μεγέθους 1, με τιμή 21,26 αφού μόνο ένας τύπος εικονικής μηχανής υπάρχει.

Στους δύο τρόπους υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών (vm placement), μεταβλητή απόφασής είναι ο πίνακας H στον πρώτο τρόπο και ο πίνακας D για τον δεύτερο

τρόπο υλοποίησης της φάσης αυτής. Ο πίνακας H είναι δυσδιάστατος με γραμμές όσες και ο αριθμός των φυσικών μηχανών και στήλες όσες και ο συνολικός αριθμός των εικονικών μηχανών u'(συνολικός αριθμός εικονικών μηχανών στον πίνακα N_i). Ο πίνακας D επίσης είναι δυσδιάστατος με γραμμές όσες και ο αριθμός των φυσικών μηχανών και στήλες όσες και αριθμός των κατηγοριών εικονικών μηχανών c. Ο πίνακας H έχει πεδίο τιμών 0 ή 1 και ο πίνακας D έχει πεδίο τιμών 0..u'(συνολικός αριθμός εικονικών μηχανών στον πίνακα N_i). Ακόμη μια μεταβλητή απόφασης και στους δυο τρόπους υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών, είναι η μεταβλητή U. Αυτή η μεταβλητή είναι ένας μονοδιάστατος πίνακας με μέγεθος όσες και οι φυσικές μηχανές και πεδίο τιμών 0 ή 1 και απεικονίζει κατά πόσο μια φυσική μηχανή είναι ενεργή ή όχι. Για τον πρώτο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών, μια φυσική μηχανή είναι ενεργή αν στον πίνακα H υπάρχει έστω και μια μονάδα στην γραμμή της συγκεκριμένης φυσικής μηχανής, δηλαδή εάν της ανατέθηκε έστω και μια εικονική μηχανή της συγκεκριμένης φυσικής μηχανής. Για τον δεύτερο τρόπο υλοποίησης, μια φυσική μηχανή είναι ενεργή αν στον πίνακα D υπάρχει έστω και ένα κελί με τιμή >0 στην γραμμή της συγκεκριμένης φυσικής μηχανής, δηλαδή εάν της ανατέθηκε έστω και μια εικονική μηχανή της συγκεκριμένης φυσικής μηχανής. Η μεταβλητή D για σκοπούς κατανόησης, στον κώδικα υλοποίησης έχει το όνομα new. Οι μεταβλητές απόφασης και στους δύο τρόπους υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών δηλώνονται ως εξής στην γλώσσα Minizinc:

```
array[pms,vms] of var 0..1:H; //Για τον πρώτο τρόπο υλοποίησης
array[pms,types] of var 0..u_vms:new; //Για τον δεύτερο τρόπο υλοποίησης
array[pms] of var 0..1:U; // Για τους δύο τρόπους υλοποίησης
```

Μετατροπή Περιορισμών για πρώτο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών στη γλώσσα Minizinc

[u είναι ο συνολικός αριθμός των εικονικών μηχανών,
q είναι ο αριθμός ο όποιος προσδιορίζει τον αριθμό των φυσικών μηχανών]

1. $\sum_{l=1}^{u} r_l^{cpu} * h_{jl} \leq C_j^{cpu}$ $\quad 1 \leq j < q$

```
constraint forall(j in pms)( sum(l in vms)(R[l,1]*H[j,l])<= Ccpu[j] );
```

2. $\sum_{l=1}^{u} r_l^{ram} * h_{jl} \leq C_j^{ram}$ $\quad 1 \leq j < q$

```
constraint forall(j in pms)(sum(l in vms)(R[l,2]*H[j,l])<= Cram[j]);
```

3. Ο περιορισμός αυτός αναφέρει ότι κάθε εικονική μηχανή θα πρέπει να ανατεθεί σίγουρα σε κάποια φυσική μηχανή, $\sum_{j=1}^{q} h_{jl} = 1$

   constraint forall(i in vms)(sum(l in pms)(H[l,i]) = 1);

4. Το διάνυσμα U προσδιορίζει κατά πόσο μια φυσική μηχανή είναι ενεργή ή όχι, δηλαδή αν τις έχουν ανατεθεί εικονικές μηχανές ή όχι. Έτσι ο περιορισμός αυτός μεταφράζεται, ως :

   constraint forall(j in pms)((exists(l in vms) (H[j,l]=1)) <-> U[j]= 1);

5. Συνάρτηση Βελτιστοποίησης

   Σκοπός είναι να ελαχιστοποιηθεί το άθροισμα $\sum_{j=1}^{q} U_j$

   solve :: int_search([H[j,l] | j in pms, l in vms], first_fail, indomain_random,complete) minimize sum(j in pms)(U[j]);

***Μετατροπή Περιορισμών για τον δεύτερο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών στη γλώσσα* Minizinc**

[c είναι αριθμός που προσδιορίζει πόσες κατηγορίες εικονικών μηχανών υπάρχουν,
q είναι ο αριθμός ο όποιος προσδιορίζει τον αριθμό των φυσικών μηχανών]

1. $\sum_{l=1}^{c} r_l^{cpu} * d_{jl} \leq C_j^{cpu}$ $\quad 1 \leq j < q$

   *constraint forall(j in pms)(sum(l in types)(R[l,1]*new[j,l])<= Ccpu[j]);*

2. $\sum_{l=1}^{c} r_l^{ram} * d_{jl} \leq C_j^{ram}$ $\quad 1 \leq j < q$

   *constraint forall(j in pms)(sum(l in types)(R[l,2]*new[j,l])<= Cram[j]);*

3. $\sum_{j=1}^{q} d_{jl}$=allocate[l], εκφράζει ότι όλες οι εικονικές μηχανές τις κατηγορίας l, θα ανατεθούν σε κάποια φυσική μηχανή(όχι απαραίτητα στην ίδια).Ο δείκτης l αναφέρεται στους τύπους εικονικών μηχανών.

   constraint forall(i in types)(sum(l in pms)(new[l,i]) = allo[i]);

4. Το διάνυσμα U αναφέρει κατά πόσο μια φυσική μηχανή είναι ενεργή ή όχι, δηλαδή αν τις έχουν ανατεθεί εικονικές μηχανές ή όχι. Έτσι ο περιορισμός αυτός μεταφράζεται , ως :

   constraint forall(j in pms)((exists(l in types) (new[j,l]>0)) <-> U[j]=1);

5. Συνάρτηση Βελτιστοποίησης

Σκοπός είναι να ελαχιστοποιηθεί το άθροισμα $\sum_{j=1}^{q} U_j$.

solve :: int_search([new[j,l] | j in pms, l in types], first_fail,indomain_random, complete) minimize sum(j in pms)(U[j]);

Η έξοδος της φάσης τοποθέτησης εικονικών μηχανών σε φυσικές μηχανές (vm placement) από τον επιλυτή Minizinc και με τους δύο τρόπους υλοποίησης είναι της μορφής, αρχικά να τυπώνεται ο ελάχιστος αριθμός φυσικών μηχανών οι οποίες θα είναι ενεργές έτσι ώστε να εξυπηρετούν όλες τις εικονικές μηχανές. Στην συνέχεια τυπώνεται η κατανομή η οποία γίνεται, δηλαδή ποιες φυσικές μηχανές είναι ενεργές και ποιες είναι απενεργοποιημένες. Επίσης τυπώνεται ο τρόπος κατανομής των εικονικών μηχανών, δηλαδή για κάθε εικονική μηχανή σε ποια φυσική μηχανή έχει ανατεθεί όσων αφορά τον πρώτο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών. Όσων αφορά τον δεύτερο τρόπος υλοποίησης τυπώνεται ο αριθμός των εικονικών που έχει ανατεθεί σε κάθε φυσική μηχανή, για κάθε τύπο εικονικών μηχανών.

6.4.3 Μορφή Αρχείου Εισόδου για τους επιλυτές Minisat+ και Bsolo

Όπως έχουμε προαναφέρει, τόσο ο επιλυτής Minisat+ όσο και ο Bsolo επιλύουν προβλήματα τα οποία αφορούν ψευδοδυαδικούς περιορισμούς. Οι επιλυτές Minisat+ και Bsolo έχουν συγκεκριμένη μορφή εισόδου για το αρχείο στο οποίο περιγράφονται οι περιορισμοί που θα πρέπει να επιλύσουν. Συγκεκριμένα θα πρέπει να γίνει κωδικοποίηση των περιορισμών του προβλήματος σε μορφή κατανοητή για τα εργαλεία Minisat+ και Bsolo. Και οι δύο επιλυτές παίρνουν ως παράμετρο εισόδου ένα αρχείο με όλους τους περιορισμούς εκφραζόμενους σε μορφή ανισοτήτων.

Η μορφή του αρχείου αυτού είναι η εξής, στην πρώτη γραμμή του αρχείου αυτού θα πρέπει να τοποθετηθεί η συνάρτηση βελτιστοποίησης. Είναι γνωστές δύο συναρτήσεις. Η συνάρτηση minimum η οποία προσπαθεί να ελαχιστοποιήσει την συνάρτηση βελτιστοποίησης, αντίστοιχα η συνάρτηση maximum η οποία προσπαθεί να μεγιστοποιήσει την συνάρτηση βελτιστοποίησης. Και οι δύο αυτές συναρτήσεις προσπαθούν να μεγιστοποιήσουν ή ελαχιστοποιήσουν μια τιμή ικανοποιώντας όσο το δυνατό περισσότερους στόχους με βάση τους περιορισμούς του προβλήματος.

Για τον επιλυτή Minisat+ η σύνταξη είναι η εξής :

Η συνάρτηση minimum για το αρχείο εισόδου έχει την μορφή :

min: -1*goal$_1$ -1*goal$_2$ …….. -1*goal$_n$;

Αντίστοιχα η συνάρτηση maximum για το αρχείο εισόδου έχει την μορφή :

max: +1*goal$_1$ +1*goal$_2$ ……. +1*goal$_n$;

Η μορφή των περιορισμών και στους δύο επιλυτές είναι στην μορφή γραμμικών ανισοτήτων. $C_0 * p_0 + C_{1*} p_1 + ... + C_{n-1*} p_{n-1} \geq C_n$ όπου οι μεταβλητές $p_0,\ldots,p_{n-1}$ ανήκουν στο πεδίο {0,1}, C_i ακέραιοι αριθμοί. Ο χαρακτήρας '-' δηλώνει ότι είναι το συμπλήρωμα (αρνητική εμφάνιση) της μεταβλητής και ο χαρακτήρας '+' δηλώνει ότι είναι η θετική εμφάνιση της μεταβλητής. Ένας όρος παίρνει την τιμή 1 εάν αποτιμάται με True (αληθές) και τιμή 0 εάν αποτιμάται με False (ψευδές). Η αριστερή πλευρά « $C_0 * p_0 + C_1 * p_1 + ... + C_{n-1} * p_{n-1}$», είναι οι όροι και στην δεξιά πλευρά είναι η σταθερά C_n. Ο συντελεστής C_i ενεργοποιείται στο πλαίσιο μιας μερικής εκχώρησης, εάν στον όρο p_i ανατεθεί η τιμή αληθείας True. Ένας ψευδοδυαδικός περιορισμός ικανοποιείται εάν ικανοποιείται η ανισότητα η οποία έχει την μορφή που προαναφέρθηκε. Συγκεκριμένα, εάν μετά την ανάθεση των τιμών το άθροισμα των συντελεστών υπερβαίνει ή είναι ίσο με την σταθερά C_n η οποία βρίσκεται στην δεξιά πλευρά της ανισότητας.

Για τον Minisat+ επιλυτή η σύνταξη των περιορισμών είναι η εξής

$C_0 * p_0 + C_{1*} p_1 + ... + C_{n-1*} p_{n-1} \geq C_n$;

π.χ : -1*x$_1$ +1*x$_2$ +1*x$_3$ ≥ 2;

Όσων αφορά την μορφή των περιορισμών και της συνάρτησης βελτιστοποίησης που δέχεται ο επιλυτής Bsolo, η σύνταξη είναι η ίδια με τον επιλυτή Minisat+ αλλά με μικρές διαφορές. Συγκεκριμένα αντί του συμβόλου '*', δέχεται κενό σύμβολο ' '.Έτσι, για αυτό το λόγο στο επόμενο υποκεφάλαιο οι περιορισμοί και η συνάρτηση βελτιστοποίησης θα αναπαριστώνται με την μορφή που δέχεται ο επιλυτής Minisat+.

6.4.4 Μετατροπή των περιορισμών του προβλήματος σε ψευδοδυαδικούς περιορισμούς για τους επιλυτές Minisat+ και Bsolo

Ο τρόπος που επιλύθηκε το πρόβλημα της αυτόνομης εικονικής διαχείρισης πόρων με τον επιλυτή Minizinc είναι με τον τρόπο που αναφέρθηκε και στο κεφάλαιο 3, δηλαδή επίλυση προβλημάτων ικανοποίησης περιορισμών(csp). Αρχικά, οριστήκαν οι μεταβλητές του προβλήματος έπειτα τα πεδία τιμών αυτών των μεταβλητών και τέλος εκφραστήκαν οι περιορισμοί στην γλώσσα Minizinc. Όπως προαναφέρθηκε οι επιλυτές Minisat+ και Bsolo επιλύουν ψευδοδυαδικούς περιορισμούς σε μορφή γραμμικών ανισοτήτων. Έτσι το πρόβλημα και οι περιορισμοί θα πρέπει να μεταφραστούν έτσι ώστε οι μεταβλητές να έχουν πεδία τιμών 0 ή 1.

Πριν προχωρήσουμε στην λεπτομερή περιγραφή για το πώς μετατράπηκε το πρόβλημα σε ψευδοδυαδικούς περιορισμούς ας αναφέρουμε ότι οι τιμές στους πίνακες που δημιουργήθηκαν στο πρόγραμμα both_phase_A.c, αποθηκεύονται σε αρχεία έτσι ώστε όταν θα δημιουργηθούν τα αρχεία εισόδου για τους επιλυτές Minisat+ και Bsolo με τους περιορισμούς της κάθε περίπτωσης εκτέλεσης να δημιουργηθούν τα ίδια δεδομένα και για τους τρεις επιλυτές με απώτερο σκοπό να γίνει η πειραματική αξιολόγησης του κεφαλαίου 7. Τα αρχεία εισόδου για κάθε ένα από τους επιλυτές Minisat+ και Bsolo δημιουργούνται από τα προγράμματα satA.c και satB.c. Το πρόγραμμα satA.c δημιουργεί τα αρχεία εισόδου για την φάση τροφοδότησης εικονικών μηχανών(vm provisioning) και για τους δύο επιλυτές. Για τον Minisat+, το αρχείο εισόδου για την φάση τροφοδότησης εικονικών μηχανών(vm provisioning) είναι το sat_inpA1m και για τον Bsolo είναι το αρχείο sat_inpA1b. Το πρόγραμμα satB.c δημιουργεί τα αρχεία εισόδου για την φάση τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές(vm placement) και για τους δύο επιλυτές. Για τον Minisat+ το αρχείο εισόδου για την φάση τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές (vm placement)για τον πρώτο τρόπο υλοποίησης είναι το sat_inpB1m και για τον δεύτερο τρόπο υλοποίησης είναι sat_inpB1_btroposm. Για τον Bsolo για τον πρώτο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών στις φυσικές μηχανές(vm placement) είναι το sat_inpB1b και για τον δεύτερο τρόπο υλοποίησης είναι sat_inpB1_btroposb.

Τώρα, ας περιγράψουμε με λεπτομέρεια πώς έγινε η μετατροπή του προβλήματος ικανοποίησης περιορισμών (csp) σε ψευδοδυαδικό πρόβλημα.

Για την φάση τροφοδότησης εικονικών μηχανών όπως έχει αναφερθεί και προηγουμένως, για την αναπαράσταση σε Minizinc η μεταβλητή απόφασης είναι ο πίνακας N_i με πεδίο τιμών από 0 μέχρι 10. Έτσι λοιπόν κάθε κελί του πίνακα αυτού θα είναι το άθροισμα μιας δεκάδας μεταβλητών οι οποίες έχουν όλες πεδίο τιμών 0 ή 1. Συγκεκριμένα, προηγουμένως είχαμε τον δυσδιάστατο πίνακα N με m γραμμές και c στήλες, ενώ τώρα έχουμε ένα πλήθος μεταβλητών, όπου για κάθε κελί του πίνακα έχουμε ένα άθροισμα 10 μεταβλητών. Δηλαδή, το κελί N[1,1] το αναπαριστούμε με τις μεταβλητές $n_{11}^1 + n_{11}^2 + n_{11}^3 + n_{11}^4 + n_{11}^5 + n_{11}^6 + n_{11}^7 + n_{11}^8 + n_{11}^9 + n_{11}^{10}$. Στην ουσία, η απεικόνιση n_{ik}^j σημαίνει ότι στο κελί n_{ik} θα αποτιμηθεί η τιμή j. Συγκεκριμένα, προσθέτουμε ένα έξτρα περιορισμό ο οποίος αναφέρει ότι $n_{ik}^j \rightarrow n_{ik}^{j-1}$ ο οποίος εξασφαλίζει ότι ένα κελί για να του αποτιμηθεί η τιμή j (δηλαδή n_{ik}^j αποτιμάται με αληθές) τότε θα πρέπει όλες οι προηγούμενες μεταβλητές στο άθροισμα των 10 μεταβλητών επίσης να είναι αποτιμούνται με αληθές. Δηλαδή, εάν γίνει η αποτίμηση n_{11}^1, n_{11}^2, n_{11}^3, n_{114}, n_{11}^5, $-n_{11}^6$, $-n_{11}^7$, $-n_{11}^8$, $-n_{11}^9$,$-n_{11}^{10}$, σημαίνει ότι το κελί n_{11} θα πάρει την τιμή 5 αφού συνολικά υπάρχουν 5 true(1). Πιο κάτω παρουσιάζεται η απεικόνιση των περιορισμών σε ψευδοδυαδικούς περιορισμούς για το αρχείο που δόθηκε ως είσοδο για την φάση τροφοδότησης εικονικών μηχανών(vm provisioning) στο υποκεφάλαιο που περιγράφτηκε η υλοποίηση σε Minizinc. Ας θυμηθούμε ποια είναι τα δεδομένα εισόδου του παραδείγματος για την φάση τροφοδότησης εικονικών μηχανών(vm provisioning).

```
m=5;   c=1; q=20;
Nmax=[|9,
    |3,
    |3,
    |2,
    |5|];
Tmax=[10,8,3,3,5];
S=[|21,26|];            // Το 21 αφορά την cpu και το 26 την ram
Ccpu=[93,49,36,31,38,74,85,32,70,98,78,89,53,41,96,37,43,43,70,50];
Cram=[44,99,44,88,55,70,88,94,34,41,70,37,55,42,87,31,56,50,56,78];
varitita=[1,2,5,3,7];
c1_sintelestis=[|3,
            |4,
            |10,
            |4,
            |3|];
c2_sintelestis=[|5,
            |3,
            |10,
            |1,
            |8|];
sintelestis_kostous=4;
```

Άρα, αφού υπάρχουν 5 εφαρμογές και 1 κατηγορία εικονικών μηχανών οι μεταβλητές θα είναι $n_{11}^{1}….n_{11}^{10}$, $n_{21}^{1}….n_{21}^{10}$, $n_{31}^{1}….n_{31}^{10}$, $n_{41}^{1}….n_{41}^{10}$, $n_{51}^{1}…..n_{51}^{10}$. Οι περιορισμοί φαίνονται πιο κάτω και όπως θα δούμε, οι ψευδοδυαδικοί περιορισμοί αποτελούν το σπάσιμο των αθροισμάτων στις περιπτώσεις που εμφανίζονται αθροίσματα σε κάποιο περιορισμό.

1. $n_{ik} \leq n_{ik}^{max}$

*1*n111 +1*n112 +1*n113 +1*n114 +1*n115 +1*n116 +1*n117 +1*n118 +1*n119 +1*n1110 <=9; // Το 9 είναι η τιμή 11 στον πίνακα n_{ikmax}*
*1*n211 +1*n212 +1*n213 +1*n214 +1*n215 +1*n216 +1*n217 +1*n218 +1*n219 +1*n2110 <=3; // Το 3 είναι η τιμή 21 στον πίνακα n_{ikmax}*
*1*n311 +1*n312 +1*n313 +1*n314 +1*n315 +1*n316 +1*n317 +1*n318 +1*n319 +1*n3110 <=3; // Το 3 είναι η τιμή 31 στον πίνακα n_{ikmax}*
*1*n411 +1*n412 +1*n413 +1*n414 +1*n415 +1*n416 +1*n417 +1*n418 +1*n419 +1*n4110 <=2; // Το 2 είναι η τιμή 41 στον πίνακα n_{ikmax}*
*1*n511 +1*n512 +1*n513 +1*n514 +1*n515 +1*n516 +1*n517 +1*n518 +1*n519 +1*n5110 <=5; // Το 3 είναι η τιμή 51 στον πίνακα n_{ikmax}*

2. $\sum_{k=1}^{c} n_{ik} \leq T_i^{max}$

1*n111 +1*n112 +1*n113 +1*n114 +1*n115 +1*n116 +1*n117 +1*n118 +1*n119 +1*n1110 <=10; // *Το 10 είναι η τιμή T_{1max}*
1*n211 +1*n212 +1*n213 +1*n214 +1*n215 +1*n216 +1*n217 +1*n218 +1*n219 +1*n2110 <=8; // *Το 8 είναι η τιμή T_{2max}*
1*n311 +1*n312 +1*n313 +1*n314 +1*n315 +1*n316 +1*n317 +1*n318 +1*n319 +1*n3110 <=3; // *Το 3 είναι η τιμή T_{3max}*
1*n411 +1*n412 +1*n413 +1*n414 +1*n415 +1*n416 +1*n417 +1*n418 +1*n419 +1*n4110 <=3; // *Το 3 είναι η τιμή T_{4max}*
1*n511 +1*n512 +1*n513 +1*n514 +1*n515 +1*n516 +1*n517 +1*n518 +1*n519 +1*n5110 <=5; // *Το 5 είναι η τιμή T_{5max}*

3. Σε κάθε εφαρμογή θα πρέπει να ανατεθεί τουλάχιστον μια εικονική μηχανή, $\sum_{k=1}^{c} n_{ik} \geq 1$.

1*n111 +1*n112 +1*n113 +1*n114 +1*n115 +1*n116 +1*n117 +1*n118 +1*n119 +1*n1110 >=1;
1*n211 +1*n212 +1*n213 +1*n214 +1*n215 +1*n216 +1*n217 +1*n218 +1*n219 +1*n2110 >=1;
1*n311 +1*n312 +1*n313 +1*n314 +1*n315 +1*n316 +1*n317 +1*n318 +1*n319 +1*n3110 >=1;
1*n411 +1*n412 +1*n413 +1*n414 +1*n415 +1*n416 +1*n417 +1*n418 +1*n419 +1*n4110 >=1;
1*n511 +1*n512 +1*n513 +1*n514 +1*n515 +1*n516 +1*n517 +1*n518 +1*n519 +1*n5110 >=1;

4. $\sum_{i=1}^{m} \sum_{k=1}^{c} n_{ik} * S_k^{cpu} \leq \sum_{j=1}^{q} C_j^{cpu}$

21*n111 +21*n112 +21*n113 +21*n114 +21*n115 +21*n116 +21*n117 +21*n118 +21*n119 +21*n1110 +21*n211 +21*n212 +21*n213 +21*n214 +21*n215 +21*n216 +21*n217 +21*n218 +21*n219 +21*n2110 +21*n311 +21*n312 +21*n313 +21*n314 +21*n315 +21*n316 +21*n317 +21*n318 +21*n319 +21*n3110 +21*n411 +21*n412 +21*n413 +21*n414 +21*n415 +21*n416 +21*n417 +21*n418 +21*n419 +21*n4110

+21*n511 +21*n512 +21*n513 +21*n514 +21*n515 +21*n516 +21*n517 +21*n518 +21*n519 +21*n5110 <=1206;

// Η τιμή 1206 είναι το άθροισμα του πίνακα
Ccpu=[93,49,36,31,38,74,85,32,70,98,78,89,53,41,96,37,43,43,70,50];
// Η τιμή 21 αφορά την ικανότητα σε Cpu της πρώτης(και μοναδικής) κατηγορίας εικονικών μηχανών

5. $\sum_{i=1}^{m}\sum_{k=1}^{c} n_{ik}*S_k^{ram} \leq \sum_{j=1}^{q} C_j^{ram}$

26*n111 +26*n112 +26*n113 +26*n114 +26*n115 +26*n116 +26*n117 +26*n118 +26*n119 +26*n1110 +26*n211 +26*n212 +26*n213 +26*n214 +26*n215 +26*n216 +26*n217 +26*n218 +26*n219 +26*n2110 +26*n311 +26*n312 +26*n313 +26*n314 +26*n315 +26*n316 +26*n317 +26*n318 +26*n319 +26*n3110 +26*n411 +26*n412 +26*n413 +26*n414 +26*n415 +26*n416 +26*n417 +26*n418 +26*n419 +26*n4110 +26*n511 +26*n512 +26*n513 +26*n514 +26*n515 +26*n516 +26*n517 +26*n518 +26*n519 +26*n5110 <=1219;

// Η τιμή 1219 είναι το άθροισμα του πίνακα
Cram=[44,99,44,88,55,70,88,94,34,41,70,37,55,42,87,31,56,50,56,78];
// Η τιμή 26 αφορά την ικανότητα σε Ram της πρώτης(και μοναδικής) κατηγορίας εικονικών μηχανών

6. Ο έξτρα περιορισμός, ο οποίος αναφέρει ότι $n_{ik}^{j} \rightarrow n_{ik}^{j-1}$ ο οποίος εξασφαλίζει ότι ένα κελί για να του αποτιμηθεί η τιμή j (δηλαδή n_{ik}^{j} αποτιμάται με αληθές) τότε θα πρέπει όλες οι προηγούμενες μεταβλητές στο άθροισμα των 10 μεταβλητών επίσης να είναι αποτιμούνται με αληθές, εκφράζεται ως εξής:
Για τις μεταβλητές $n_{11}^{1}..n_{11}^{10}$, δηλαδή $n_{11}^{j} \rightarrow n_{11}^{j-1}$ είναι:

-1*n112 +1*n111 >=0;
-1*n113 +1*n112 >=0;
-1*n114 +1*n113 >=0;
-1*n115 +1*n114 >=0;
-1*n116 +1*n115 >=0;
-1*n117 +1*n116 >=0;
-1*n118 +1*n117 >=0;
-1*n119 +1*n118 >=0;
-1*n1110 +1*n119 >=0;

Το ίδιο γράφεται και για τις υπόλοιπες μεταβλητές $n_{21}^{1}..n_{21}^{10}$, $n_{31}^{1}..n_{31}^{10}$, $n_{41}^{1}..n_{41}^{10}$, $n_{51}^{1}..n_{51}^{10}$.

7. Συνάρτηση Βελτιστοποίησης

Uglobal= maximize $(10\text{-}e) * \sum_{i=1}^{m}(a_i*U_i) - (e*total) * \sum_{i=1}^{m}(Cost_i)$

Για την συνάρτησης Βελτιστοποίησης έχουν διαχωριστεί τα δυο αθροίσματα που παρατηρούμε, και μετατράπηκε σε συνάρτησης ελαχιστοποίησης και είναι η εξής:

Uglobal= minimize $(e*total) * \sum_{i=1}^{m}(Cost_i) - (10\text{-}e) * \sum_{i=1}^{m}(a_i*U_i)$

min:70*n111 +70*n112 +70*n113 +70*n114 +70*n115 +70*n116 +70*n117 +70*n118 +70*n119 +70*n1110 +42*n211 +42*n212 +42*n213 +42*n214 +42*n215 +42*n216 +42*n217 +42*n218 +42*n219 +42*n2110 +140*n311 +140*n312 +140*n313 +140*n314 +140*n315 +140*n316 +140*n317 +140*n318 +140*n319 +140*n3110 +14*n411 +14*n412 +14*n413 +14*n414 +14*n415 +14*n416 +14*n417 +14*n418 +14*n419 +14*n4110 +112*n511 +112*n512 +112*n513 +112*n514 +112*n515 +112*n516 +112*n517 +112*n518 +112*n519 +112*n5110 -24*n111 -24*n112 -24*n113 -24*n114 -24*n115 -24*n116 -24*n117 -24*n118 -24*n119 -24*n1110 -64*n211 -64*n212 -64*n213 -64*n214 -64*n215 -64*n216 -64*n217 -64*n218 -64*n219 -64*n2110 -80*n311 -80*n312 -80*n313 -80*n314 -80*n315 -80*n316 -80*n317 -80*n318 -80*n319 -80*n3110 -96*n411 -96*n412 -96*n413 -96*n414 -96*n415 -96*n416 -96*n417 -96*n418 -96*n419 -96*n4110 -0*n511 -0*n512 -0*n513 -0*n514 -0*n515 -0*n516 -0*n517 -0*n518 -0*n519 -0*n5110 ;

Η πιο πάνω απεικόνιση συμπεριλαμβάνει το σπάσιμο της εξίσωσης U_i η οποία αναφέρεται στο επίπεδο των πόρων για την εφαρμογή α_i και ορίζεται ως $U_i=f_i(N_i)= c1_{i1}*n_{i1} + c1_{i2}*n_{i2} + ... + c1_{im}*n_{im}$ και του λειτουργικού κόστος για την εφαρμογή α_i το οποίο ορίζεται ως $Cost_i=c2_{i1}*n_{i1} + c2_{i2}*n_{i2} + ... + c2_{im}*n_{im}$.

Για την φάση τοποθέτησης εικονικών μηχανών (vm placement), μεταβλητές απόφασης είναι ο πίνακας U και ο πίνακας H για τον πρώτο τρόπο υλοποίησης και ο πίνακας D για τον δεύτερο τρόπο υλοποίησης. Όσων αφορά τον πίνακα U, μετατράπηκε σε τόσες μεταβλητές όσο και το μέγεθος του. Συγκεκριμένα υπάρχουν οι μεταβλητές $u_1,u_2,\ldots,u_q$ (q - αριθμός φυσικών μηχανών).Η απεικόνιση u_1 σημαίνει ότι η φυσική μηχανή 1 είναι ενεργή και αντίστοιχα η απεικόνιση $-u_1$ σημαίνει ότι η φυσική μηχανή u1 είναι μη ενεργή. Συνεπώς η συνάρτηση βελτιστοποίησης της φάσης αυτής κωδικοποιείται και στους δυο τρόπους υλοποίησης ως εξής:

*min:1*u1+1*u2+1*u3+…+1*uq;*

Τώρα ας δούμε πως μετατρέπεται η απεικόνιση των μεταβλητών H και D.

Για τον πρώτο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών, το διάνυσμα $H_j=(h_{j1},h_{j2},h_{jl},\ldots,h_{ju'})$ συμβολίζει το σύνολο των εικονικών μηχανών που ανατίθενται στην φυσική μηχανή p_j. Τυπικά, η τοποθέτηση εικονικών μηχανών σε φυσικές μηχανές συμβολίζεται ως $H_{jl}=1$ εάν η φυσική μηχανή p_j φιλοξενεί την εικονική μηχανή vml. Το H_j είναι δυαδικό διάνυσμα (0 ή 1). Όπως έχει αναφερθεί προηγουμένως, ο πίνακας H είναι δυσδιάστατος με γραμμές όσες και ο αριθμός των φυσικών μηχανών και στήλες όσες και ο συνολικός αριθμός των εικονικών μηχανών u'(συνολικός αριθμός εικονικών μηχανών στον πίνακα N_i) και έχει πεδίο τιμών 0 ή 1. Έτσι για να αναπαρασταθεί ως ψευδοδυαδικό πρόβλημα θα πρέπει για κάθε κελί του πίνακα H να υπάρχει μια μεταβλητή. Δηλαδή υπάρχουν οι μεταβλητές $h_{11},h_{12},..,h_{qu}$'(q απεικονίζει τον αριθμό των φυσικών μηχανών, u'

απεικονίζει τον συνολικό αριθμό των εικονικών μηχανών που απαιτούνται) και η αποτίμηση h_{ij} σημαίνει ότι η φυσική μηχανή i φιλοξένει την εικονική μηχανή j, αντίστοιχα $-h_{ij}$ σημαίνει ότι η φυσική μηχανή i δεν φιλοξένει την εικονική μηχανή j. Για να εξασφαλίσουμε τον περιορισμό ότι η φυσική μηχανή i είναι ενεργή αν υπάρχει έστω και μια μονάδα h_{ij} στην γραμμή της φυσικής μηχανής i, δηλαδή εάν της ανατέθηκε έστω και μια εικονική μηχανή προσθέτουμε τον γραμμικό περιορισμό $u_i-h_{ij}\geq 0$. Πιο κάτω παρουσιάζεται η απεικόνιση των περιορισμών ως ψευδοδυαδικοί περιορισμοί ,για το αρχείο που δόθηκε ως είσοδο για την φάση τοποθέτησης εικονικών μηχανών(vm placement) με τον πρώτο τρόπο υλοποίησης στο υποκεφάλαιο που περιγράφτηκε η υλοποίηση σε Minizinc. Ας θυμηθούμε ποια είναι τα δεδομένα εισόδου του παραδείγματος για την φάση τοποθέτησης εικονικών μηχανών με τον πρώτο τρόπο υλοποίησης.

```
q=20; //Αριθμός φυσικών μηχανών
Ccpu=[93,49,36,31,38,74,85,32,70,98,78,89,53,41,96,37,43,43,70,50];
Cram=[44,99,44,88,55,70,88,94,34,41,70,37,55,42,87,31,56,50,56,78];
u_vms=8; // Ο αριθμός u΄-συνολικός αριθμός εικονικών μηχανών που απαιτούνται
R=[|21,26,
  |21,26,
  |21,26,
  |21,26,
  |21,26,
  |21,26,
  |21,26|];
```

Άρα αφού υπάρχουν 20 φυσικές μηχανές και 8 εικονικές μηχανές οι μεταβλητές για την επίλυση σε ψευδοδυαδικούς περιορισμούς θα είναι h_{11},h_{12}..,h_{18} για την πρώτη φυσική μηχανή και h_{201}..h_{208} για την 20-στη φυσική μηχανή όπως επίσης u_1 μέχρι u_{20} για την αναπαράσταση αν μια φυσική μηχανή θα είναι ενεργή ή όχι. Οι περιορισμοί φαίνονται πιο κάτω και όπως θα δούμε, οι ψευδοδυαδικοί περιορισμοί αποτελούν το σπάσιμο των αθροισμάτων στις περιπτώσεις τις οποίες εμφανίζονται αθροίσματα σε κάποιο περιορισμό.

1. $\sum_{l=1}^{u} r_l^{cpu} * h_{jl} \leq C_j^{cpu}$ $1 \leq j < q.$

```
21*h11+21*h12+21*h13+21*h14+21*h15+21*h16+21*h17+21*h18<=93;
21*h21+21*h22+21*h23+21*h24+21*h25+21*h26+21*h27+21*h28<=49;
21*h31+21*h32+21*h33+21*h34+21*h35+21*h36+21*h37+21*h38<=36;
21*h41+21*h42+21*h43+21*h44+21*h45+21*h46+21*h47+21*h48<=31;
21*h51+21*h52+21*h53+21*h54+21*h55+21*h56+21*h57+21*h58<=38;
21*h61+21*h62+21*h63+21*h64+21*h65+21*h66+21*h67+21*h68<=74;
21*h71+21*h72+21*h73+21*h74+21*h75+21*h76+21*h77+21*h78<=Ccpu7;
                              …
                              …
21*h201+21*h202+21*h203+21*h204+21*h205+21*h206+21*h207+21*h208<=50;
```

2. $\sum_{l=1}^{u} r_l^{ram} * h_{jl} \leq C_j^{ram}$ $1 \leq j < q$

```
26*h11+26*h12+26*h13+26*h14+26*h15+26*h16+26*h17+26*h18<=44;
26*h21+26*h22+26*h23+26*h24+26*h25+26*h26+26*h27+26*h28<=99;
26*h31+26*h32+26*h33+26*h34+26*h35+26*h36+26*h37+26*h38<=44;
26*h41+26*h42+26*h43+26*h44+26*h45+26*h46+26*h47+26*h48<=88;
26*h51+26*h52+26*h53+26*h54+26*h55+26*h56+26*h57+26*h58<=55;
26*h61+26*h62+26*h63+26*h64+26*h65+26*h66+26*h67+26*h68<=70;
26*h71+26*h72+26*h73+26*h74+26*h75+26*h76+26*h77+26*h78<=Cram7;
                        …
                        …
26*h201+26*h202+26*h203+26*h204+26*h205+26*h206+26*h207+26*h208<=78;
```

3. Ο περιορισμός αυτός αναφέρεται στο ότι κάθε εικονική μηχανή θα πρέπει να ανατεθεί σίγουρα σε κάποια φυσική μηχανή, $\sum_{j=1}^{q} h_{jl} = 1$

```
1*h11+1*h21+1*h31+1*h41+1*h51+1*h61+1*h71+1*h81+1*h91+1*h101+1*h111+1*h121+1*
h131+1*h141+1*h151+1*h161+1*h171+1*h181+1*h191+1*h201=1;
1*h12+1*h22+1*h32+1*h42+1*h52+1*h62+1*h72+1*h82+1*h92+1*h102+1*h112+1*h122+1*
h132+1*h142+1*h152+1*h162+1*h172+1*h182+1*h192+1*h202=1;
                        …
                        …

1*h18+1*h28+1*h38+1*h48+1*h58+1*h68+1*h78+1*h88+1*h98+1*h108+1*h118+1*h128+1*
h138+1*h148+1*h158+1*h168+1*h178+1*h188+1*h198+1*h208=1;
```

4. Το διάνυσμα U αναφέρει κατά πόσο μια φυσική μηχανή είναι ενεργή ή όχι, δηλαδή αν τις έχουν ανατεθεί εικονικές μηχανές ή όχι. Για να εξασφαλίσουμε τον περιορισμό ότι η φυσική μηχανή i είναι ενεργή αν υπάρχει έστω και μια μονάδα h_{ij} στην γραμμή της φυσικής μηχανής i προσθέτουμε τον γραμμικό περιορισμό $u_i - h_{ij} \geq 0$. Αυτό μετατρέπεται ως:

Για την πρώτη φυσική μηχανή :

```
1*u1-1*h11>=0;
1*u1-1*h12>=0;
1*u1-1*h13>=0;
1*u1-1*h14>=0;
1*u1-1*h15>=0;
1*u1-1*h16>=0;
1*u1-1*h17>=0;
1*u1-1*h18>=0;
```

….

Για την 20-οστή φυσική μηχανή :

```
1*u20-1*h201>=0;
1*u20-1*h202>=0;
1*u20-1*h203>=0;
1*u20-1*h204>=0;
1*u20-1*h205>=0;
1*u20-1*h206>=0;
1*u20-1*h207>=0;
1*u20-1*h208>=0;
```

5. Συνάρτηση Βελτιστοποίησης

   Σκοπός είναι να ελαχιστοποιηθεί το άθροισμα $\sum_{j=1}^{q} U_j$

   min:1*u1+1*u2+1*u3+1*u4+1*u5+1*u6+1*u7+1*u8+1*u9+1*u10+1*u11+1*u12+1*u13+1*u14+1*u15+1*u16+1*u17+1*u18+1*u19+1*u20;

Για τον δεύτερο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών, το διάνυσμα $D_j=(d_{j1},d_{j2},h_{jl},\ldots,h_{jc})$, συμβολίζει το σύνολο των εικονικών μηχανών που ανατίθενται στην φυσική μηχανή p_j. Τυπικά, η τοποθέτηση εικονικής μηχανής σε φυσική μηχανή συμβολίζεται ως $D_{jl} >1$ εάν p_j φιλοξενεί έστω και μια εικονική μηχανή τύπου l. Προφανώς, το D_j δεν είναι δυαδικό διάνυσμα (0 ή 1) αλλά έχει πεδίο τιμών από 0 μέχρι u', τον συνολικό αριθμό εικονικών μηχανών που απαιτούνται(το άθροισμα του διανύσματος N_i). Ο πίνακας D επίσης είναι δυσδιάστατος με γραμμές όσες και ο αριθμός των φυσικών μηχανών(q) και στήλες όσες και αριθμός των κατηγοριών εικονικών μηχανών(c). Έτσι λοιπόν κάθε κελί του πίνακα αυτού θα είναι το άθροισμα από u' μεταβλητές οι οποίες έχουν όλες πεδίο τιμών 0 ή 1. Συγκεκριμένα, προηγουμένως είχαμε τον δυσδιάστατο πίνακα D με q γραμμές και c στήλες ενώ τώρα έχουμε ένα πλήθος μεταβλητών, όπου για κάθε κελί του πίνακα έχουμε ένα άθροισμα από u' μεταβλητές. Δηλαδή, το κελί D[1,1] το αναπαριστούμε με τις μεταβλητές $d_{11}^{1}+d_{11}^{2}+..+d_{11}^{u'}$. Στην ουσία, η αναπαράσταση d_{ik}^{j} σημαίνει ότι το κελί d_{ik} θα πάρει την τιμή j(δηλαδή της φυσικής μηχανής i θα της ανατεθούν j εικονικές μηχανές τύπου k). Συγκεκριμένα, προσθέτουμε ένα έξτρα περιορισμό ο οποίος λέει ότι $d_{ik}^{j} \rightarrow d_{ik}^{j-1}$ ο οποίος εξασφαλίζει ότι ένα κελί για να του αποτιμηθεί η τιμή j (δηλαδή d_{ik}^{j} αποτιμάται με αληθές) τότε θα πρέπει όλες οι προηγούμενες στο άθροισμα των u' μεταβλητών επίσης να είναι αποτιμούνται με αληθές. Δηλαδή εάν γίνει η αποτίμηση d_{11}^{1}, d_{11}^{2}, d_{11}^{3},- d_{11}^{4},- d_{11}^{5}, - $d_{11}^{u'}$, σημαίνει ότι το κελί d_{11} θα πάρει την τιμή 3 αφού έχω συνολικά 3 true(1), άρα της φυσικής μηχανής 1 θα της ανατεθούν 3 εικονικές μηχανές τύπου 1. Για να εξασφαλίσουμε τον περιορισμό ότι η φυσική μηχανή i είναι ενεργή εάν στον πίνακα D υπάρχει έστω και ένα κελί με τιμή $d_{ij}>0$ για την γραμμή της συγκεκριμένης φυσικής μηχανής i, δηλαδή εάν της ανατέθηκε έστω και μια εικονική μηχανή, προσθέτουμε τον γραμμικό περιορισμό $u_i- d_{ik}^{j}\geq 0$. Πιο κάτω παρουσιάζεται η απεικόνιση των περιορισμών ως ψευδοδυαδικοί περιορισμοί, για το αρχείο που δόθηκε ως είσοδο για την φάση τοποθέτησης εικονικών μηχανών με τον δεύτερο τρόπο υλοποίησης στο υποκεφάλαιο που περιγράφτηκε η υλοποίηση σε Minizinc. Ας θυμηθούμε ποια είναι τα δεδομένα εισόδου του παραδείγματος για την φάση τοποθέτησης εικονικών μηχανών με τον δεύτερο τρόπο υλοποίησης.

q=20; // Αριθμός φυσικών μηχανών
Ccpu=[93,49,36,31,38,74,85,32,70,98,78,89,53,41,96,37,43,43,70,50];
Cram=[44,99,44,88,55,70,88,94,34,41,70,37,55,42,87,31,56,50,56,78];
u_vms=8; /Συνολικός αριθμός εικονικών μηχανών που απαιτούνται
type_vms=1; // αριθμός κατηγοριών εικονικών μηχανών
allo=[8]; // διάνυσμα allocate
R=[|21,26,|];

Άρα, αφού υπάρχουν 20 φυσικές μηχανές και 8 εικονικές μηχανές από την κατηγορία 1, οι μεταβλητές θα είναι $d_{11}^{1}..d_{11}^{8}$,$d_{21}^{1}..d_{21}^{8}$ μέχρι $d_{201}^{1}..d_{201}^{8}$. Οι περιορισμοί φαίνονται πιο κάτω και όπως θα δούμε, οι ψευδοδυαδικοί περιορισμοί θα είναι το σπάσιμο των αθροισμάτων, αν εμφανίζονται αθροίσματα σε κάποιο περιορισμό.

Πιο κάτω, αντί της απεικόνισης d_{ik}^{j} θα χρησιμοποιείται η απεικόνιση h_{ik}^{j} για τους ψευδοδυαδικούς περιορισμούς.

1. $\sum_{l=1}^{c} r_l^{cpu} * d_{jl} \leq C_j^{cpu}$ $\quad 1 \leq j < q$

```
21*h111+21*h112+21*h113+21*h114+21*h115+21*h116+21*h117+21*h118<=93;
+21*h211+21*h212+21*h213+21*h214+21*h215+21*h216+21*h217+21*h218<=49;
+21*h311+21*h312+21*h313+21*h314+21*h315+21*h316+21*h317+21*h318<=36;
+21*h411+21*h412+21*h413+21*h414+21*h415+21*h416+21*h417+21*h418<=31;
+21*h511+21*h512+21*h513+21*h514+21*h515+21*h516+21*h517+21*h518<=38;
+21*h611+21*h612+21*h613+21*h614+21*h615+21*h616+21*h617+21*h618<=74;
+21*h711+21*h712+21*h713+21*h714+21*h715+21*h716+21*h717+21*h718<=Ccpu7;
                                    …
                                    …
+21*h2011+21*h2012+21*h2013+21*h2014+21*h2015+21*h2016+21*h2017+21*h2018<=50;
```

2. $\sum_{l=1}^{c} r_l^{ran} * d_{jl} \leq C_j^{ran}$ $\quad 1 \leq j < q$

```
26*h111+26*h112+26*h113+26*h114+26*h115+26*h116+26*h117+26*h118<=44;
+26*h211+26*h212+26*h213+26*h214+26*h215+26*h216+26*h217+26*h218<=99;
+26*h311+26*h312+26*h313+26*h314+26*h315+26*h316+26*h317+26*h318<=44;
+26*h411+26*h412+26*h413+26*h414+26*h415+26*h416+26*h417+26*h418<=88;
+26*h511+26*h512+26*h513+26*h514+26*h515+26*h516+26*h517+26*h518<=55;
+26*h611+26*h612+26*h613+26*h614+26*h615+26*h616+26*h617+26*h618<=70;
+26*h711+26*h712+26*h713+26*h714+26*h715+26*h716+26*h717+26*h718<=Cram7;
                                    …
                                    …
+26*h2011+26*h2012+26*h2013+26*h2014+26*h2015+26*h2016+26*h2017+26*h2018<=78;
```

3. $\sum_{j=1}^{q} d_{jl}$=allocate[l], εκφράζει ότι όλες οι εικονικές μηχανές για την κατηγορία l, θα ανατεθούν σε κάποια φυσική μηχανή(όχι απαραίτητα στην ίδια). Ο δείκτης l αναφέρεται στους τύπους εικονικών μηχανών.

```
1*h111+1*h112+1*h113+1*h114+1*h115+1*h116+1*h117+1*h118+ // 1η φυσική μηχανή
1*h211+1*h212+1*h213+1*h214+1*h215+1*h216+1*h217+1*h218+//2η φυσική μηχανή
```

…..+
1*h2011+1*h2012+1*h2013+1*h2014+1*h2015+1*h2016+1*h2017+1*h2018=8;// $20^{η}$

Στο συγκεκριμένο παράδειγμα είναι μόνο ένας περιορισμός αφού μόνο μια κατηγορία εικονικών μηχανών υπάρχει.

4. Το διάνυσμα U αναφέρει κατά πόσο μια φυσική μηχανή είναι ενεργή ή όχι, δηλαδή αν τις έχουν ανατεθεί εικονικές μηχανές ή όχι. Για να εξασφαλίσουμε τον περιορισμό ότι η φυσική μηχανή i είναι ενεργή αν υπάρχει έστω και μια μονάδα h_{ij} στην γραμμή της φυσικής μηχανής i, δηλαδή εάν την ανατέθηκε έστω και μια εικονική μηχανή προσθέτουμε τον γραμμικό περιορισμό $u_i - h_{ik}^{j} \geq 0$. Αυτό μετατρέπεται ως:

Για την πρώτη φυσική μηχανή : 1*u1-1*h111>=0;
1*u1-1*h112>=0;
1*u1-1*h113>=0;
1*u1-1*h114>=0;
1*u1-1*h115>=0;
1*u1-1*h116>=0;
1*u1-1*h117>=0;
1*u1-1*h118>=0;

Για την 20-οστή φυσική μηχανή : 1*u20-1*h2011>=0;
1*u20-1*h2012>=0;
1*u20-1*h2013>=0;
1*u20-1*h2014>=0;
1*u20-1*h2015>=0;
1*u20-1*h2016>=0;
1*u20-1*h2017>=0;
1*u20-1*h2018>=0;

5. Για τον έξτρα περιορισμό ο οποίος λέει ότι $d_{ik}^{j} \rightarrow d_{ik}^{j-1}$ ο οποίος εξασφαλίζει ότι ένα κελί για να του αποτιμηθεί η τιμή j (δηλαδή d_{ik}^{j} αποτιμάται με αληθές) τότε θα πρέπει όλες οι προηγούμενες στο άθροισμα των u' μεταβλητών επίσης να είναι αποτιμούνται με αληθές. Αυτό μεταφράζεται ως:
Για τις μεταβλητές $h_{11}^{1}..h_{11}^{8}$, δηλαδή $h_{11}^{j} \rightarrow h_{11}^{j-1}$ είναι :

1*h111-1*h112>=0;
1*h112-1*h113>=0;
1*h113-1*h114>=0;
1*h114-1*h115>=0;
1*h115-1*h116>=0;
1*h116-1*h117>=0;
1*h117-1*h118>=0;

Το ίδιο γράφεται και για τις υπόλοιπες μεταβλητές μέχρι την $h_{201}^{1}..h_{201}^{8}$.

6. Συνάρτηση Βελτιστοποίησης
Σκοπός είναι να ελαχιστοποιηθεί το άθροισμα $\sum_{j=1}^{q} U_j$

```
min:1*u1+1*u2+1*u3+1*u4+1*u5+1*u6+1*u7+1*u8+1*u9+1*u10+1*u11+1*u12+1*u13+1*u1
4+1*u15+1*u16+1*u17+1*u18+1*u19+1*u20;
```

Η είσοδος για την φάση τοποθέτησης εικονικών μηχανών προς επίλυση του προβλήματος με ψευδοδυαδικούς περιορισμούς στους επιλυτές Minisat+ και Bsolo παίρνεται από την κατανομή που κάνει ο επιλυτής Minizinc, επειδή τα αποτελέσματα της κατανομής της φάσης τροφοδότησης εικονικών μηχανών για το πόσες εικονικές μηχανές απαιτούνται είναι τα ίδια και για τους τρείς επιλυτές Minizinc, Minisat+ και Bsolo. Αυτό γίνεται, δεδομένου ότι είναι δυσανάλογα επίπονη η διαδικασία μετατροπής, ιδιαίτερα λαμβανομένου υπόψη του σκοπού της παρούσας διπλωματικής, ο οποίος δεν είναι η συγκεκριμένη μετατροπή καθεαυτή.

6.5 Τρόπος Εκτέλεσης Προγραμμάτων

Πιο κάτω παρουσιάζονται τα βήματα εκτέλεσης των προγραμμάτων που έχουν υλοποιηθεί. Για να δημιουργηθούν τυχαίες τιμές για τους πίνακες Nmax, Tmax, Ccpu, Cram, S, c1_sintelestis. c2_sintelestis. και στην συνέχεια να εκτελέσουμε τους τρείς επιλυτές εκτελούμε τα ακόλουθα βήματα:

1. gcc both_phase_A.c –o both_phase_A
   ./both_phase_A
2. time minizinc 1omeros.mzn --output-to-file in.txt A_inputA1.dzn
3. gcc satA.c –o sat
   ./satA
4. ./minisat+ sat_inpA1m
5. ./bsolo_pb10 sat_inpA1b
6. gcc both_phase_B.c –o both_phase_B
   ./both_phase_B
7. time minizinc 2o_meros_B.mzn B_inputB1.dzn –all-solutions
8. time minizinc 2o_btropos.mzn B_inputB1_btropos.dzn –all-solutions
9. gcc satB.c –o satB
   ./satB
10. ./minisat+ sat_inpB1m
11. ./minisat+ sat_inpB1_btroposm
12. ./ bsolo_pb10 sat_inpB1b
13. ./ bsolo_pb10 sat_inpB1_btroposb

Κεφάλαιο 7

Πειραματική Ανάλυση και Συμπεράσματα

Στο κεφάλαιο αυτό, αρχικά θα παρουσιαστούν τα πειράματα που έχουν πραγματοποιηθεί. Στη συνέχεια με βάση τα αποτελέσματα των πειραμάτων θα παρουσιαστούν τα συμπεράσματα τα οποία έχουν προκύψει. Τέλος, θα γίνει μια μικρή αναφορά στην μελλοντική επέκταση της παρούσας διπλωματικής εργασίας.

7.1 **Πειραματική Ανάλυση**

Αρχικά πραγματοποιήθηκαν κάποια πειράματα για τη φάση τροφοδότησης εικονικών μηχανών(vm provisioning), με στόχο να παρατηρηθεί πως τα δεδομένα εισόδου του χρήστη επηρεάζουν τον χρόνο εκτέλεσης της φάσης αυτής για κάθε επιλυτή, καθώς επίσης και την συνάρτηση βελτιστοποίησης της φάσης αυτής.

Όπως έχουμε αναφέρει και προηγουμένως σκοπός αυτής της φάσης είναι να γίνει μια κατανομή στο διάνυσμα Ni για κάθε εφαρμογή και να προσδιοριστεί πόσες εικονικές μηχανές από κάθε κατηγορία χρειάζεται η κάθε εφαρμογή. Αυτή η κατανομή θα πρέπει να γίνει με τέτοιο τρόπο έτσι ώστε να μεγιστοποιείται η συνάρτηση ικανοποίησης(Global Utility Function) η οποία δίνει ένα μέτρο ικανοποίησης των εφαρμογών.

Τα δεδομένα εισόδου από τον χρήστη είναι ο αριθμός των εφαρμογών(m), ο αριθμός των κατηγοριών των εικονικών μηχανών(c), ο αριθμός των φυσικών μηχανών(q), ο συντελεστής κόστους(e) και η βαρύτητα για κάθε εφαρμογή.

Περισσότερη έμφαση θα δοθεί στο πώς επηρεάζει ο αριθμός των κατηγοριών εικονικών μηχανών, ο συντελεστής κόστους και οι αριθμοί των βαρυτήτων των εφαρμογών, τον χρόνο εκτέλεσης αλλά και την τιμή της συνάρτησης βελτιστοποίησης για την φάση τροφοδότησης των εικονικών μηχανών.

Στην συνέχεια, όπως θα δούμε πιο κάτω, έχουν γίνει ολοκληρωμένα πειράματα τα οποία αφορούν ολόκληρο το πρόβλημα, δηλαδή και τις δύο φάσεις τροφοδότησης και τοποθέτησης εικονικών μηχανών. Σκοπός των πειραμάτων αυτών είναι να συγκρίνουμε τον χρόνο εκτέλεσης και ολοκλήρωσης ενός ολοκληρωμένου προβλήματος στο κάθε επιλυτή αλλά να δούμε και την ποιότητα της λύσης του κάθε επιλυτή. Αφού, όπως θα δούμε στην συνέχεια είναι δυνατόν ένας επιλυτής να τερματίζει και να βρίσκει την βέλτιστη λύση για το πόσες φυσικές μηχανές θα πρέπει να είναι ενεργές ενώ κάποιος άλλος επιλυτής να μην μπορεί να τερματίσει και να σταματά σε ένα σημείο εύρεσης της βέλτιστης λύσης όπου και η λύση του δεν είναι η βέλτιστη στο σημείο αυτό. Τέλος, θα δούμε πως επηρεάζεται ο χρόνος εκτέλεσης της φάσης τοποθέτησης εικονικών μηχανών(Vm placement) από την διαφορά μεταξύ των εικονικών μηχανών και των φυσικών μηχανών.

Στα πειράματα που έχουν πραγματοποιηθεί πιο κάτω, δοκιμάστηκαν οι τρεις επιλυτές με τυχαίες, αυθαίρετες τιμές για τα διανύσματα Nmax, Tmax, Ccpu, Cram, S, c1_sintelestis, c2_sintelestis. Όπως έχουμε προαναφέρει τα δεδομένα εισόδου από τον χρήστη είναι ο αριθμός των εφαρμογών(m), ο αριθμός των κατηγοριών των εικονικών μηχανών(c), ο αριθμός των φυσικών μηχανών(q), ο συντελεστής κόστους(e) και η βαρύτητα της κάθε εφαρμογής. Έτσι, για το πρώτο μέρος της πειραματικής αξιολόγησης το οποίο αφορά πειράματα για την φάση τροφοδότησης εικονικών μηχανών, δοκιμάστηκε ο μέγιστος αριθμός

για τον οποίο θεωρούμε ότι και οι τρεις επιλυτές βγάζουν ικανοποιητικά αποτελέσματα από άποψη χρόνου, συγκεκριμένα δοκιμάστηκε ο αριθμός 12 για τις εφαρμογές αφού είναι και ο μέγιστος αριθμός εφαρμογών στον οποίο ο επιλυτής Minizinc αντέχει. Όσων αφορά το δεύτερο μέρος της πειραματικής αξιολόγησης το οποίο αφορά ολοκληρωμένα πειράματα για τις φάσεις τροφοδότησης και τοποθέτησης εικονικών μηχανών, δοκιμάστηκαν οι τιμές από 5 μέχρι 12 εφαρμογές και 2 μέχρι 4 κατηγορίες εικονικών μηχανών. Έτσι, αρχικά δοκιμάσαμε τους επιλυτές σε μικρά δεδομένα εισόδου και στην συνέχεια σταδιακά μεγάλωναν οι τιμές για τις εφαρμογές και τις κατηγορίες εικονικών μηχανών μέχρι το μέγιστο που αντέχουν οι επιλυτές.

7.1.1 Μέρος Α - Πειράματα για τη φάση Τροφοδότησης Εικονικών Μηχανών(VM Provisioning)

Αρχικά έχουν γίνει 12 πειράματα για την φάση τροφοδότησης εικονικών μηχανών(vm provisioning), με σταθερό τον αριθμό των εφαρμογών σε 12 και αλλάζοντας κάθε φορά τον αριθμό των κατηγοριών εικονικών μηχανών, τον συντελεστή κόστους και τις βαρύτητες για τις εφαρμογές. Συγκεκριμένα τα πιο κάτω πειράματα είναι για 2 και 3 κατηγορίες εικονικών μηχανών και κάθε φορά αλλάζει ο αριθμός για τον συντελεστή κόστους και οι αριθμοί των βαρυτήτων των εφαρμογών.

Πιο κάτω παρουσιάζονται 3 πίνακες, ο κάθε ένας από αυτούς αναφέρεται στους επιλυτές Minizinc, Minisat+ και Bsolo αντίστοιχα. Στους πιο κάτω πίνακες, στην πρώτη στήλη απεικονίζεται ο αύξον αριθμός του πειράματος, την $2^{η}$ – $6^{η}$ στήλη αποτελούν τα δεδομένα εισόδου από τον χρήστη, στην $7^{η}$ στήλη απεικονίζεται ο αριθμός των εικονικών μηχανών που αποφασίζει ο επιλυτής, στην $8^{η}$ φαίνεται η συνάρτηση βελτιστοποίησης, δηλαδή η ικανοποίηση του μοντέλου για το συγκεκριμένο παράδειγμα. Στην τελευταία στήλη, απεικονίζεται ο χρόνος που χρειάζεται ο επιλυτής για τον οποίο γίνεται αναφορά στον συγκεκριμένο πίνακα. Η τελευταία στήλη η οποία αφορά τον χρόνο εκτέλεσης του κάθε επιλυτή, στις περιπτώσεις όπου υπάρχει το σύμβολο «*» μετά τον χρόνο εκτέλεσης σημαίνει ότι δεν έχει τερματίσει ο συγκεκριμένος επιλυτής και η λύση στη στήλη της συνάρτησης βελτιστοποίησης είναι η τιμή μέχρι την δεδομένη στιγμή που σκόπιμα τερματίστηκε το πρόγραμμα.

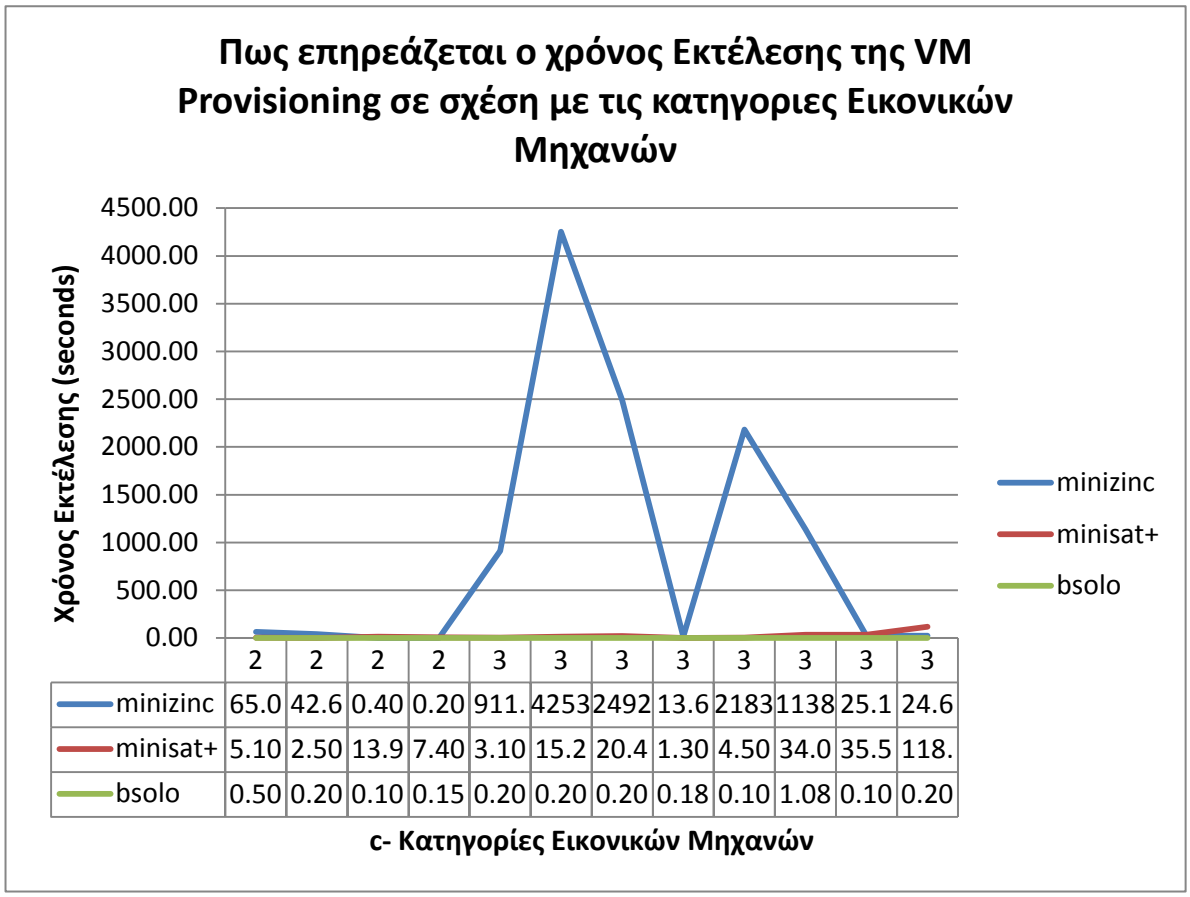
Minizinc - ΦάσηΤροφοδότησης Εικονικών Μηχανών (VM Provisioning)								
Αρ.Πειράματος	Παράμετροι							
	m - Αριθμός Εφαρμογών	c – Κατηγορίες VM	Συντελεστής Κόστους	Βαρύτητα Εφαρμογών	q - Αριθμός PM	u- Αριθμός vms	Global utility function	Χρόνος (minutes) vm_provisioning
1	12	2	1	1,0,0,1,2,0,3,1,0,2,1,4	15	26	-1350	1:05.29
2	12	2	4	3,4,6,0,1,0,7,2,0,1,3,0	12	16	3072	0:42.57
3	12	2	7	5,6,9,2,1,3,0,4,0,6,0,1	15	12	14166	0:00.44
4	12	2	9	7,4,3,8,9,10,2,6,3,4,5,8	18	12	37496	0:00.15
5	12	3	1	1,0,0,1,2,0,0,0,1,0,2,1	14	30	-1352	15:11.87
6	12	3	2	0,1,0,2,0,1,3,0,2,1,1,0	15	28	-396	1:10:53:37 (*)
7	12	3	2	1,0,5,2,3,0,0,4,1,0,1,2	13	31	-2808	41:31.72
8	12	3	2	0,0,0,1,0,0,2,0,0,1,2,0	16	30	-324	0:13.55
9	12	3	3	0,2,4,6,1,1,0,0,3,6,4,1	18	16	1715	36:22:86
10	12	3	3	1,6,7,9,9,6,3,2,5,0,1,4	20	18	2241	18:57:91
11	12	3	5	4,5,7,0,2,3,9,4,5,8,0,1	25	17	4820	0:25.11
12	12	3	8	7,8,6,4,10,3,9,2,0,1,4,0	22	12	22176	0:24.63

Minisat+ - ΦάσηΤροφοδότησης Εικονικών Μηχανών (VM Provisioning)								
Αρ.Πειράματος	Παράμετροι							
	m - Αριθμός Εφαρμογών	c – Κατηγορίες VM	Συντελεστής Κόστους	Βαρύτητα Εφαρμογών	q - Αριθμός PM	u- Αριθμός vms	Global utility function	Χρόνος (minutes) vm_provisioning
1	12	2	1	1,0,0,1,2,0,3,1,0,2,1,4	15	26	-1350	0:05.13
2	12	2	4	3,4,6,0,1,0,7,2,0,1,3,0	12	16	3072	0:02.53
3	12	2	7	5,6,9,2,1,3,0,4,0,6,0,1	15	12	14166	0:13.85
4	12	2	9	7,4,3,8,9,10,2,6,3,4,5,8	18	12	37496	0:07.36
5	12	3	1	1,0,0,1,2,0,0,0,1,0,2,1	14	30	-1352	0:03.10
6	12	3	2	0,1,0,2,0,1,3,0,2,1,1,0	15	28	-420	0:15.20
7	12	3	2	1,0,5,2,3,0,0,4,1,0,1,2	13	31	-2808	0:20.35
8	12	3	2	0,0,0,1,0,0,2,0,0,1,2,0	16	30	-324	0:01.29
9	12	3	3	0,2,4,6,1,1,0,0,3,6,4,1	18	16	1715	0:04.52
10	12	3	3	1,6,7,9,9,6,3,2,5,0,1,4	20	18	2241	0:33.98
11	12	3	5	4,5,7,0,2,3,9,4,5,8,0,1	25	17	4820	0:35.46
12	12	3	8	7,8,6,4,10,3,9,2,0,1,4,0	22	12	22176	1:58.24

Bsolo - ΦάσηΤροφοδότησης Εικονικών Μηχανών (VM Provisioning)								
Αρ.Πειράματος	Παράμετροι							
	m - Αριθμός Εφαρμο-γών	c – Κατηγο-ρίες VM	Συντελεστής Κόστους	Βαρύτητα Εφαρμογών	q - Αριθμός PM	u- Αριθμός vms	Global utility function	Χρόνος (minutes) vm_provisioning
1	12	2	1	1,0,0,1,2,0,3,1,0,2,1,4	15	26	-1350	0:00.05
2	12	2	4	3,4,6,0,1,0,7,2,0,1,3,0	12	16	3072	0:00.02
3	12	2	7	5,6,9,2,1,3,0,4,0,6,0,1	15	12	14166	0:00.01
4	12	2	9	7,4,3,8,9,10,2,6,3,4,5,8	18	12	37496	0:00.15
5	12	3	1	1,0,0,1,2,0,0,0,1,0,2,1	14	30	-1352	0:00.02
6	12	3	2	0,1,0,2,0,1,3,0,2,1,1,0	15	28	-420	0:00.02
7	12	3	2	1,0,5,2,3,0,0,4,1,0,1,2	13	31	-2808	0.:00.02
8	12	3	2	0,0,0,1,0,0,2,0,0,1,2,0	16	30	-324	0:00.18
9	12	3	3	0,2,4,6,1,1,0,0,3,6,4,1	18	16	1715	0:00.01
10	12	3	3	1,6,7,9,9,6,3,2,5,0,1,4	20	18	2241	0:01.08
11	12	3	5	4,5,7,0,2,3,9,4,5,8,0,1	25	17	4820	0:00.01
12	12	3	8	7,8,6,4,10,3,9,2,0,1,4,0	22	12	22176	0:00.02

- Γραφική αναπαράσταση για το πώς επηρεάζεται ο χρόνος εκτέλεσης

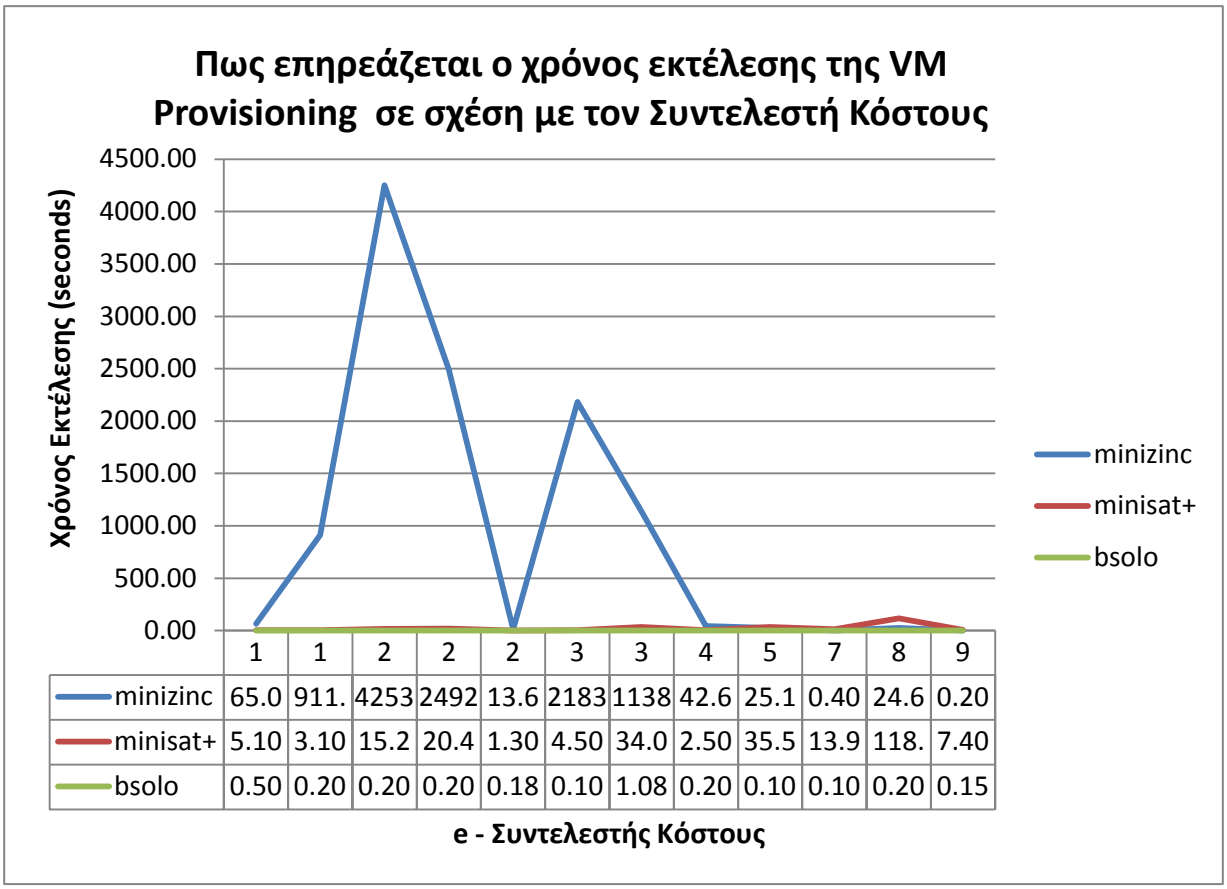
1. Σ ε σχέση με τις κατηγορίες εικονικών μηχανών



	2	2	2	2	3	3	3	3	3	3	3	3
minizinc	65.0	42.6	0.40	0.20	911.	4253	2492	13.6	2183	1138	25.1	24.6
minisat+	5.10	2.50	13.9	7.40	3.10	15.2	20.4	1.30	4.50	34.0	35.5	118.
bsolo	0.50	0.20	0.10	0.15	0.20	0.20	0.20	0.18	0.10	1.08	0.10	0.20

Όπως παρατηρούμε από την πιο πάνω γραφική παράσταση, η αύξηση των κατηγοριών εικονικών μηχανών(c), πολλές φορές επηρεάζει τον χρόνο εκτέλεσης της φάσης τροφοδότησης εικονικών μηχανών(vm provisioning) στον επιλυτή Minizinc κυρίως. Παρατηρούμε διακυμάνσεις στο χρόνο εκτέλεσης της φάσης αυτής στο επιλυτή Minizinc αφού τις περισσότερε φορές αυξάνεται ραγδαία καθώς αυξάνονται οι κατηγορίες εικονικών μηχανών, αλλά και μερικές φορές η αύξηση στον χρόνο δεν είναι τόσο εμφανής. Για τους άλλους δυο επιλυτές Minisat+ και Bsolo είναι εμφανές ότι η αύξηση των κατηγοριών εικονικών μηχανών δεν επηρεάζει σχεδόν καθόλου τον χρόνο εκτέλεσης τους, μόνο στο επιλυτή Minisat+ βλέπουμε μια ελάχιστη αύξηση στα τελευταία πειράματα. Παρόλα αυτά η αύξηση αυτή δεν είναι εμφανές και μπορεί να θεωρηθεί αμελητέα.

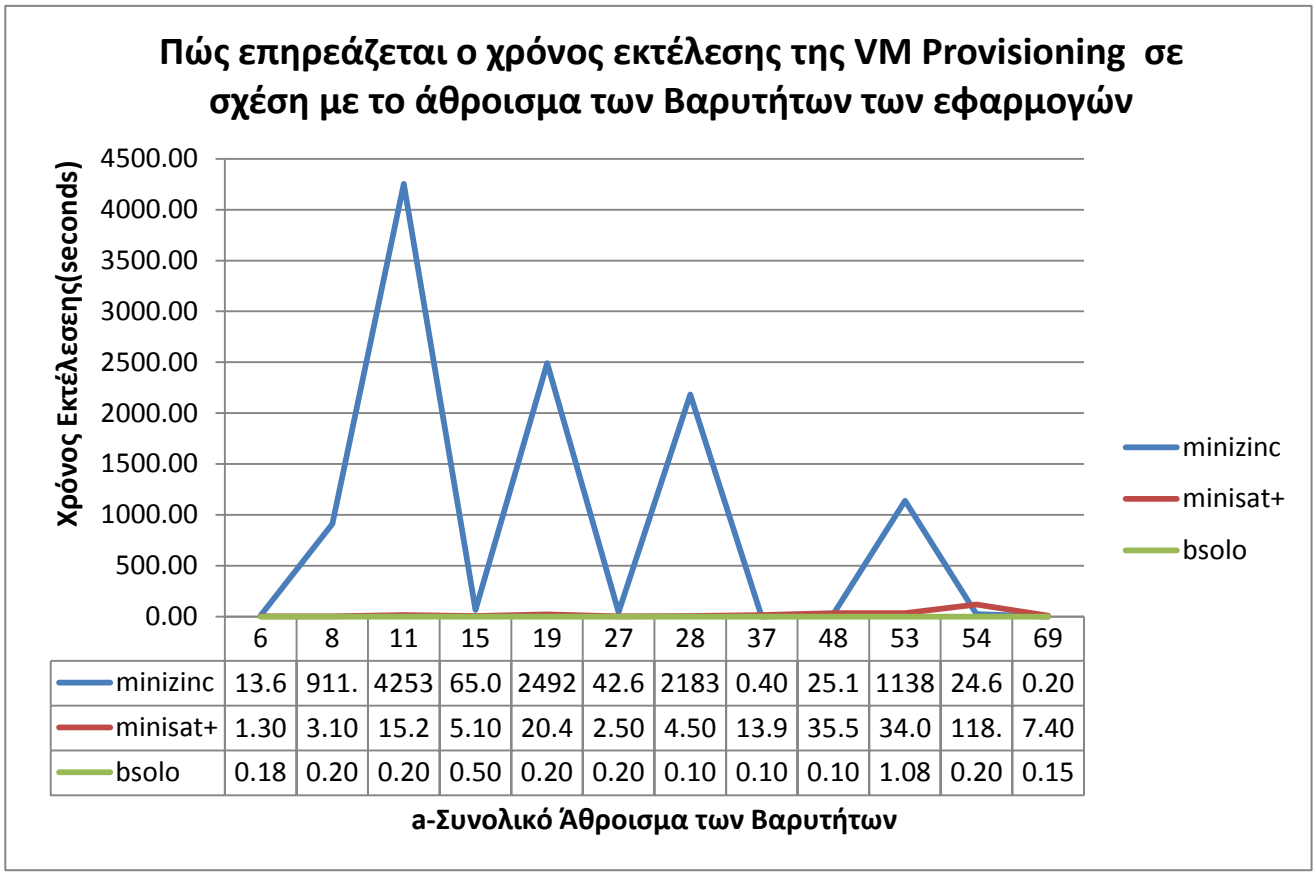
2. Σε σχέση με τον συντελεστή κόστους



	1	1	2	2	2	3	3	4	5	7	8	9
minizinc	65.0	911.	4253	2492	13.6	2183	1138	42.6	25.1	0.40	24.6	0.20
minisat+	5.10	3.10	15.2	20.4	1.30	4.50	34.0	2.50	35.5	13.9	118.	7.40
bsolo	0.50	0.20	0.20	0.20	0.18	0.10	1.08	0.20	0.10	0.10	0.20	0.15

Με βάση την πιο πάνω γραφική παράσταση, παρατηρούμε διακυμάνσεις στο χρόνο εκτέλεσης της φάσης αυτής στον επιλυτή Minizinc αφού μερικές φορές αυξάνεται πολύ αλλά και μερικές φορές η αύξηση στον χρόνο δεν είναι τόσο εμφανής. Συγκεκριμένα, βλέπουμε ότι η αύξηση στον χρόνο εκτέλεσης του Minizinc γίνεται στα σημεία όπου ο συντελεστής κόστους δεν είναι μεγάλος αριθμός. Ακόμη βλέπουμε ότι για μεγαλύτερους αριθμούς στον συντελεστή κόστους όπως φαίνεται στην γραφική για τους αριθμούς 4, 5, 7, 8, 9 ο χρόνος εκτέλεσης στον Minizinc μειώνεται αρκετά και γίνεται πολύ μικρός. Για τους άλλους δυο επιλυτές Minisat+ και Bsolo είναι εμφανές ότι η αύξηση του συντελεστή κόστους δεν επηρεάζει σχεδόν καθόλου τον χρόνο εκτέλεσης τους. Σημαντικό είναι το γεγονός ότι στις περιπτώσεις όπου ο συντελεστής κόστους είναι μεγάλος αριθμός (7,8,9), ο επιλυτής Minisat+ είναι λίγο πιο αργός σε σχέση με τον Minizinc. Φυσικά στις περιπτώσεις αυτές η διαφορά του χρόνου στους δύο αυτούς επιλυτές είναι ελάχιστη.

3. Σε σχέση με το συνολικό Άθροισμα των Βαρυτήτων

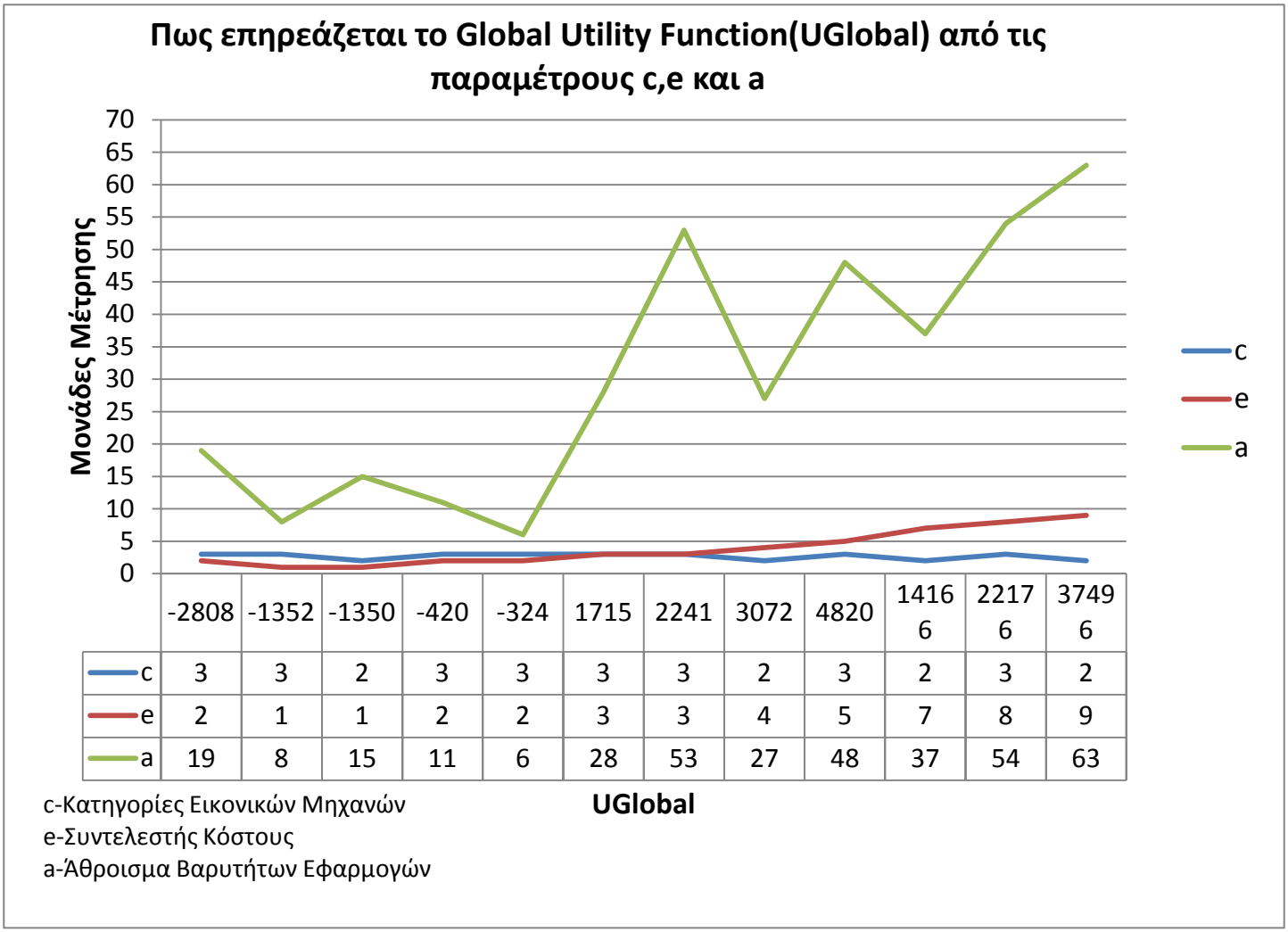


	6	8	11	15	19	27	28	37	48	53	54	69
minizinc	13.6	911.	4253	65.0	2492	42.6	2183	0.40	25.1	1138	24.6	0.20
minisat+	1.30	3.10	15.2	5.10	20.4	2.50	4.50	13.9	35.5	34.0	118.	7.40
bsolo	0.18	0.20	0.20	0.50	0.20	0.20	0.10	0.10	0.10	1.08	0.20	0.15

Στην πιο πάνω γραφική απεικονίζεται η επίδραση του χρόνου εκτέλεσης της φάσης τροφοδότησης εικονικών μηχανών από το άθροισμα των βαρυτήτων των εφαρμογών. Όπως έχει αναφερθεί και προηγουμένως, στην φάση τροφοδότησης εικονικών μηχανών, ο χρήστης δίνει ως δεδομένο εισόδου έναν ακέραιο αριθμό για κάθε εφαρμογή, ο αριθμός αυτός αναφέρεται στην βαρύτητα της κάθε εφαρμογής. Έτσι, το άθροισμα των βαρυτήτων είναι η πρόσθεση των βαρυτήτων των εφαρμογών που δίνει ο χρήστης σε μια εκτέλεση του προγράμματος(σε ένα πείραμα). Με βάση την πιο πάνω γραφική παράσταση, η αύξηση του αθροίσματος των βαρυτήτων κάθε πειράματος, πολλές φορές επηρεάζει τον χρόνο εκτέλεσης της φάσης τροφοδότησης εικονικών μηχανών(vm provisioning) στον επιλυτή Minizinc. Παρατηρούμε πολλές διακυμάνσεις στον χρόνο εκτέλεσης της φάσης αυτής στον επιλυτή Minizinc αφού αρκετές φορές αυξάνεται πολύ. Συγκεκριμένα, βλέπουμε ότι η μεγαλύτερη αύξηση στον χρόνο εκτέλεσης του Minizinc γίνεται στα αρχικά σημεία όπου το συνολικό άθροισμα των βαρυτήτων δεν είναι μεγάλος αριθμός. Ακόμη βλέπουμε ότι καθώς αυξάνεται το συνολικό άθροισμα των βαρυτήτων η αύξηση στον χρόνο εκτέλεσης του Minizinc τις περισσότερες φορές μειώνεται, συγκεκριμένα μειώνεται ο ρυθμός αύξησης του χρόνου εκτέλεσης. Για τους άλλους δυο επιλυτές, Minisat+ και Bsolo είναι εμφανές ότι η αύξηση

συνολικού αθροίσματος των βαρυτήτων των εφαρμογών δεν επηρεάζει σχεδόν καθόλου τον χρόνο εκτέλεσης τους. Σημαντικό είναι το γεγονός ότι στις περιπτώσεις όπου το άθροισμα των βαρυτήτων είναι μεγάλος αριθμός (54,69), ο Minisat+ είναι λίγο πιο αργός σε σχέση με τον επιλυτή Minizinc.

- Γραφική Αναπαράσταση για το πώς επηρεάζεται η συνάρτηση UGlobal



	-2808	-1352	-1350	-420	-324	1715	2241	3072	4820	14166	22176	37496
c	3	3	2	3	3	3	3	2	3	2	3	2
e	2	1	1	2	2	3	3	4	5	7	8	9
a	19	8	15	11	6	28	53	27	48	37	54	63

UGlobal

c-Κατηγορίες Εικονικών Μηχανών
e-Συντελεστής Κόστους
a-Άθροισμα Βαρυτήτων Εφαρμογών

Η συνάρτηση ικανοποίησης των εφαρμογών (Global Utility Function - UGlobal) όπως προαναφέρθηκε είναι το μέτρο ικανοποίησης της κάθε εφαρμογής. Στην πιο πάνω γραφική βλέπουμε πως το UGlobal επηρεάζεται σε σχέση με την αύξηση των κατηγοριών εικονικών μηχανών, του συντελεστή κόστους και του συνολικού αθροίσματος των βαρυτήτων. Στον κατακόρυφο άξονα απεικονίζονται οι τιμές του συντελεστή κόστους, των κατηγοριών εικονικών μηχανών και του συνολικού αθροίσματος των βαρυτήτων των εφαρμογών σε σχέση με την τιμή της συνάρτησης ικανοποίησης των εφαρμογών(UGlobal) στον οριζόντιο άξονα. Οι τιμές αυτές ονομάστηκαν μονάδες μέτρησης για τον λόγο ότι είναι ακέραιοι

αριθμοί. Είναι εμφανές ότι η αύξηση των βαρυτήτων άλλα και του συντελεστή κόστους επηρεάζουν την αύξηση της συνάρτησης αυτής. Συγκεκριμένα, βλέπουμε ότι καθώς ο συντελεστής κόστους αυξάνεται, αυξάνεται και το UGlobal. Επίσης στην γραφική η οποία απεικονίζει τις βαρύτητες των εφαρμογών, βλέπουμε πολλές διακυμάνσεις εντούτοις είναι προφανές ότι καθώς αυξάνεται το άθροισμα των βαρυτήτων αυξάνεται και η τιμή του UGlobal. Σημαντικό είναι το γεγονός ότι η αύξηση των κατηγοριών εικονικών μηχανών δεν επηρεάζει σχεδόν καθόλου την τιμή του UGlobal. Έτσι, συμπεραίνουμε ότι στην μεγιστοποίηση της συνάρτηση ικανοποίησης των εφαρμογών UGlobal καθοριστικός είναι ο ρόλος του συντελεστή κόστους και των βαρυτήτων των εφαρμογών.

7.1.2 Μέρος Β - Ολοκληρωμένα Πειράματα για τις φάσεις, Τροφοδότησης Εικονικών Μηχανών(VM Provisioning) και Τοποθέτησης Εικονικών Μηχανών στις Φυσικές Μηχανές(VM Placement)

Στη συνέχεια έχουν γίνει 38 πειράματα τα οποία αφορούν ολοκληρωμένα προβλήματα για την αυτόνομη και δυναμική διαχείριση των πόρων, δηλαδή τα πειράματα αυτά αφορούν τις δύο φάσεις τροφοδότησης και τοποθέτησης εικονικών μηχανών (vm provisioning και vm placement).

Τα πειράματα αυτά ξεκινούν από 5 εφαρμογές και φθάνουν μέχρι 12 εφαρμογές. Για 5 μέχρι 9 εφαρμογές δοκιμάστηκαν οι τιμές 2 και 3 για τις κατηγορίες εικονικών μηχανών και για 10 μέχρι 12 εφαρμογές δοκιμάστηκαν οι τιμές 2,3 και 4 για τις κατηγορίες εικονικών μηχανών. Για κάθε περίπτωση αριθμού εικονικών μηχανών δοκιμάστηκαν δύο περιπτώσεις για τον αριθμό των φυσικών μηχανών με σκοπό να δοκιμάσουμε τους επιλυτές σε δύσκολα προβλήματα και σε προβλήματα μεσαίας δυσκολίας όσων αφορά την φάση τοποθέτησης εικονικών μηχανών σε φυσικές μηχανές έτσι ώστε να ελαχιστοποιηθεί ο αριθμός των ενεργών φυσικών μηχανών.

Η είσοδος για την φάση τοποθέτησης εικονικών μηχανών προς επίλυση του προβλήματος με ψευδοδυαδικούς περιορισμούς στους επιλυτές Minisat+ και Bsolo παίρνεται από την κατανομή που κάνει ο επιλυτής Minizinc στη φάση τροφοδότησης εικονικών μηχανών (vm provisioning), επειδή τα αποτελέσματα της κατανομής της φάσης τροφοδότησης εικονικών μηχανών για το πόσες εικονικές μηχανές απαιτούνται είναι τα ίδια και για τους τρείς επιλυτές Minizinc, Minisat+ και Bsolo. Αυτό γίνεται δεδομένου ότι είναι δυσανάλογα επίπονη η διαδικασία μετατροπής, ιδιαίτερα λαμβανομένου υπόψη του σκοπού της παρούσας διπλωματικής, ο οποίος δεν είναι η συγκεκριμένη μετατροπή καθεαυτή.

Πιο κάτω παρουσιάζονται 3 πίνακες, ο κάθε ένας από αυτούς αναφέρεται στους επιλυτές Minizinc, Minisat+ και Bsolo αντίστοιχα. Στους πιο κάτω πίνακες, στην πρώτη στήλη απεικονίζεται ο αύξον αριθμός του πειράματος, οι στήλες $2^{η}$ – $4^{η}$ αποτελούν τα δεδομένα εισόδου του χρήστη (αριθμός εφαρμογών(m), αριθμός κατηγοριών εικονικών μηχανών(c), αριθμός φυσικών μηχανών(q)), στην $5^{η}$ στήλη βρίσκεται ο αριθμός των εικονικών μηχανών που αποφασίζει ο κάθε επιλυτής κατά την φάση τροφοδότησης εικονικών μηχανών(vm provisioning). Οι στήλες 6 και 7 αφορούν την φάση τροφοδότησης. Η στήλη 6 απεικονίζει την συνάρτηση βελτιστοποίησης(UGlobal) και η στήλη 7 τον χρόνος εκτέλεσης της φάσης τροφοδότησης που χρειάζεται ο επιλυτής για τον οποίο γίνεται αναφορά στον συγκεκριμένο πίνακα. Οι στήλες $8^{η}$-$11^{η}$ αφορούν την φάση τοποθέτησης. Οι στήλες 8 και 9 αναφέρονται στον πρώτο τρόπο εκτέλεσης της φάσης τοποθέτησης εικονικών μηχανών και συγκεκριμένα στον αριθμό των ενεργών φυσικών μηχανών(active pms) και στον χρόνο εκτέλεσης του πρώτου τρόπου για τον συγκεκριμένο επιλυτή στον οποίο αναφέρεται ο συγκεκριμένος πίνακας, αντίστοιχα. Τέλος, οι στήλες 10 και 11 αναφέρονται στον δεύτερο τρόπο εκτέλεσης την φάσης τοποθέτησης εικονικών μηχανών και συγκεκριμένα στον αριθμό των ενεργών φυσικών μηχανών(active pms) και στον χρόνο εκτέλεσης του δεύτερου τρόπου για τον συγκεκριμένο επιλυτή στον οποίο αναφέρεται ο συγκεκριμένος πίνακας, αντίστοιχα.

Οι στήλες 7, 9 και 11 οι οποίες αφορούν τον χρόνο εκτέλεσης του κάθε επιλυτή, στις περιπτώσεις όπου υπάρχει το σύμβολο «*» μετά τον χρόνο εκτέλεσης, σημαίνει ότι δεν έχει τερματίσει ο συγκεκριμένος επιλυτής και η λύση στη στήλη της συνάρτησης βελτιστοποίησης(6,8,10 στήλες αντίστοιχα), είναι η τιμή μέχρι την δεδομένη στιγμή που σκόπιμα τερματίστηκε το πρόγραμμα στα 10 λεπτά. Επίσης, στις περιπτώσεις όπου υπάρχει το σύμβολο «-» μετά τον χρόνο εκτέλεσης σημαίνει ότι ο επιλυτής έχει βρει την επιθυμητή βέλτιστη λύση και δεν έχει τερματίσει και ο χρόνος εκτέλεσης που αναγράφεται είναι ο χρόνος μέχρι να βρει την επιθυμητή βέλτιστη λύση.

Minizinc - Ολοκληρωμένα Πειράματα, Τροφοδότηση και Τοποθέτηση Εικονικών Μηχανών										
Αρ.Πειράματος	Παράμετροι				1η Φάση- Τροφοδότησης Εικονικών Μηχανών		2η Φάση- Τοποθέτησης Εικονικών Μηχανών			
							1ος Τρόπος		2ος Τρόπος	
	m - Αριθμός Εφαρμο-γών	c – Κατηγορίες VM	q - Αριθμός PM	u- Αριθμός vms	UGlobal	Χρόνος (minutes)	Active Pms	Χρόνος (minutes)	Active Pms	Χρόνος(minutes)
1	5	2	10	9	-683	0:00.57	4	0:24.35	4	0:00.10
2	5	2	12	7	-52	0:00.15	3	0:01.06	3	0:00.24
3	5	3	12	18	-1070	0:01.15	9	4:02.95 (-)	9	0:02.39
4	5	3	16	14	-360	0:01.30	4	10:56.47	4	0:05.48
5	6	2	16	18	-504	0:00.67	10	9:33:47 (*)	8	0:38.04
6	6	2	18	14	-211	0:00.15	6	9:76:21 (*)	5	1:51.61
7	6	3	18	25	-1837	0:00.89	7	11:33.12 (*)	9	10:00.09 (*)
8	6	3	22	15	-468	0:00.35	6	9:59:42 (*)	5	3:12.22 (-)
9	7	2	18	19	-364	0:00.20	8	9:32.12 (*)	6	3:17.23 (-)
10	7	2	15	19	-180	0:00.30	7	10:01.22 (*)	6	1:19.001 (-)
11	7	3	18	22	-180	0:04.69	12	10:08.67 (*)	13	9:24.33 (*)
12	7	3	15	15	-824	0:04.96	4	5:45:23 (-)	4	0:13.88
13	8	2	15	21	-232	0:03.54	8	9:43:03 (*)	7	0:14.41
14	8	2	18	24	-1912	0:00.13	15	9:12:39 (*)	14	3:12:52 (-)
15	8	3	15	21	-808	0:01.15	6	9:28:82 (*)	6	10:16.33 (*)
16	8	3	18	28	-914	0:51.01	10	10:02.31 (*)	10	9.53.34 (*)
17	9	2	15	16	-134	0:12.34	5	9:32.33 (*)	4	0.01.68
18	9	2	18	14	-248	0:00.51	4	9:02.33 (-)	4	0:03.57

19	9	3	15	13	-48	0:59.84	3	0:38.02	3	0:01.02
20	9	3	18	21	-1212	0:06.70	11	7:45.21 (-)	11	2:49.52 (-)
21	10	2	18	19	-130	0:19.71	10	4:33.10 (-)	10	1:11.49
22	10	2	20	22	-828	0:00.29	8	10:02.33 (*)	7	10:31.36 (*)
23	10	3	18	18	-134	0:01.05	7	9:38.12 (*)	7	10:02.33 (*)
24	10	3	15	14	-360	0:00.29	4	9:42.19 (*)	3	0:01.92
25	10	4	15	22	-656	0:20.56	12	9:54.14 (*)	10	3:46.23 (-)
26	10	4	18	28	-1124	0:20.57	13	12:02.32 (*)	13	9:44.02 (*)
27	11	2	15	21	-412	0:01.59	8	9:53.12 (*)	6	1:57.16
28	11	2	18	13	740	0:20.97	3	3:49 08	3	0:00.34
29	11	3	18	18	-252	0:11.39	8	9:53.36 (*)	7	1:56.39 (-)
30	11	3	15	27	-1126	0:17.00	14	10:02.33 (*)	11	2:33.14 (-)
31	11	4	18	23	-324	0:13.23	9	9:56.48 (*)	9	9:45.32 (*)
32	11	4	22	19	-832	0:04.63	7	10:00.03 3 (*)	9	10:11.12 (*)
33	12	2	20	19	226	0:01.19	9	9:59.03 (*)	7	2:44.12 (-)
34	12	2	18	24	300	0:53.17	10	10:00.03 (*)	10	9:49.48 (*)
35	12	3	17	19	-66	0:27.29	7	1:12.45 (-)	8	10:02.13 (*)
36	12	3	20	21	-332	0:02.68	9	10:00.08 (*)	9	9:42.56 (*)
37	12	4	18	19	-38	1:03.44	9	10:43.35 (*)	7	9:53.58 (*)
38	12	4	15	24	-616	0:22.53	12	10:58.42 (*)	11	6:53.39 (-)

Minisat+ - Ολοκληρωμένα Πειράματα, Τροφοδότηση και Τοποθέτηση Εικονικών Μηχανών										
Αρ.Πειράματος	Παράμετροι				1η Φάση- Τροφοδότησης Εικονικών Μηχανών		2η Φάση- Τοποθέτησης Εικονικών Μηχανών			
							1ος Τρόπος		2ος Τρόπος	
	m - Αριθμός Εφαρμο-γών	c – Κατηγο-ρίες VM	q - Αριθμός PM	u- Αριθμός vms	UGlobal	Χρόνος (minutes)	Active Pms	Χρόνος (minutes)	Active Pms	Χρόνος(minutes)
1	5	2	10	9	-683	0:00:13	4	0:00.115	4	0:00.0267
2	5	2	12	7	-52	0:00.065	3	0:00.009	3	0:00.0417
3	5	3	12	18	-1070	0:00.795	9	11:21.33 *	9	0:00..1767
4	5	3	16	14	-360	0:00.2583	4	0:71.11	4	0:00.0121
5	6	2	16	18	-504	0:00.69	8	8:31:29 (-)	8	0:00.616
6	6	2	18	14	-211	0:00.22	5	4:32:21 (-)	5	0:01.41
7	6	3	18	25	-1837	0:01.90	6	6:33.42 (-)	6	0:28.58
8	6	3	22	15	-468	0:00.528	5	3:33:69 (-)	5	0:04.68
9	7	2	18	19	-364	0:00.283	6	1:53.62 (-)	6	0:04.16
10	7	2	15	19	-180	0:00.28	6	1:63.20 (-)	6	0:00.63
11	7	3	18	22	-180	0:00.845	11	5:22.34 (-)	11	1:11.37
12	7	3	15	15	-824	0:02.55	4	0:07.73	4	0:00.63
13	8	2	15	21	-232	0:01.52	7	1:33.26 (-)	7	0:00.215
14	8	2	18	24	-1912	0:00.94	14	1:00.03 (-)	14	0:24.36
15	8	3	15	21	-808	0:00.598	6	9:36:54 (*)	5	0:02.80
16	8	3	18	28	-914	0:03.94	9	9:15:21 (*)	9	9:00.03 (*)

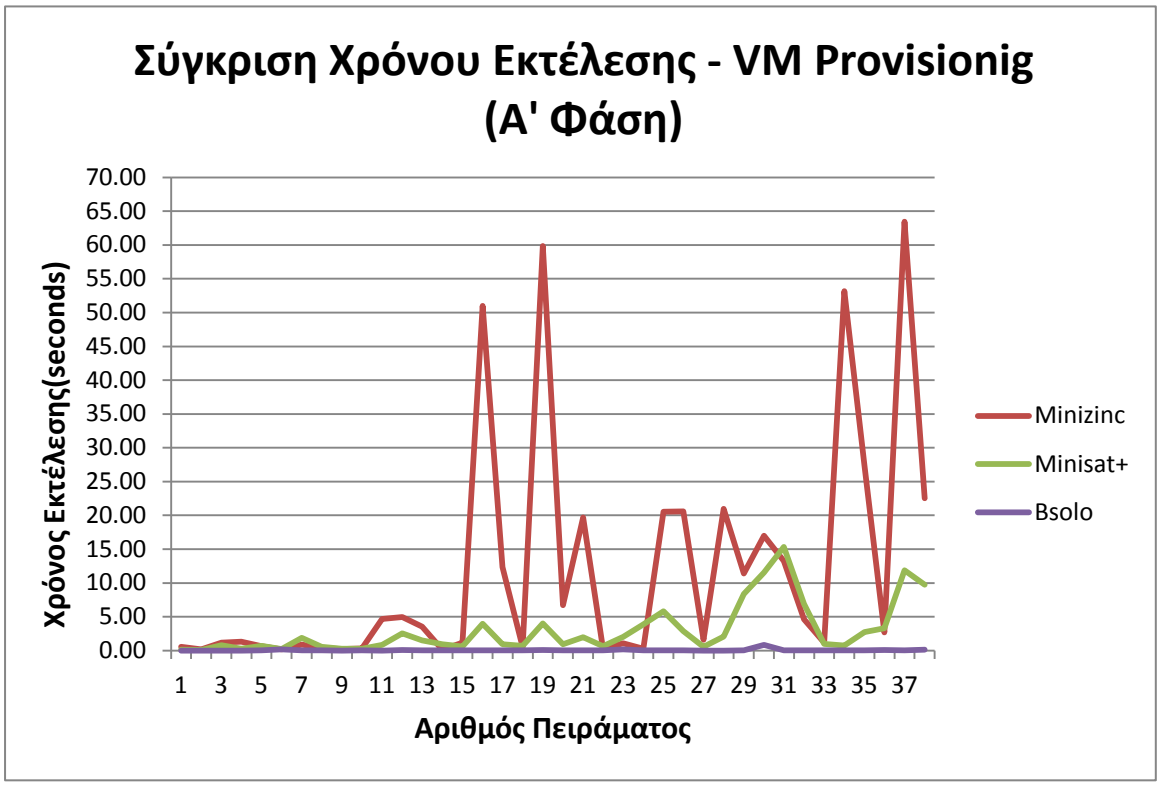
17	9	2	15	16	-134	0:00.92	4	0:39.73	4	0:00.29
18	9	2	18	14	-248	0:00.69	4	0:18.48	4	0:00.49
19	9	3	15	13	-48	0:04.00	3	0:00.93	3	0:00.20
20	9	3	18	21	-1212	0:00.94	11	2:42.02 (-)	11	0:00.42
21	10	2	18	19	-130	0:01.98	10	2:00.02 (-)	10	0:00.86
22	10	2	20	22	-828	0:00.67	6	1:33.02 (-)	6	0:06.33
23	10	3	18	18	-134	0:02.04	6	2:02.08 (-)	6	0:00.52
24	10	3	15	14	-360	0:03.87	3	0:00.71	3	0:00.39
25	10	4	15	22	-656	0:05.82	10	2:33.49 (-)	10	0:08.97
26	10	4	18	28	-1124	0:02.89	10	4:44.21 (-)	10	3:19.62
27	11	2	15	21	-412	0:00.56	6	1:02.33 (-)	6	0:01.003
28	11	2	18	13	740	0:02.07	3	0:00.35	3	0:00.18
29	11	3	18	18	-252	0:08.35	7	4:12.33 (-)	7	0:33.20
30	11	3	15	27	-1126	0:11.53	11	3:32.36 (-)	11	0:40.67
31	11	4	18	23	-324	0:15.32	8	9:53.32 (*)	8	10:02.33 (*)
32	11	4	22	19	-832	0:06.92	7	10:00.06 (*)	7	10:00.29 (*)
33	12	2	20	19	226	0:00.99	7	1:34.12 (-)	7	0:01.198
34	12	2	18	24	300	0:00.73	10	9:59.42 (*)	9	0:07.84
35	12	3	17	19	-66	0:02.71	7	1:00.03 (-)	7	0:06.19
36	12	3	20	21	-332	0:03.30	8	1:00.04 (-)	8	0:27.62
37	12	4	18	19	-38	0:11.88	6	6:12.37 (-)	6	0:10.88
38	12	4	15	24	-616	0:09.73	11	1:03.27 (-)	11	0:64.85

Bsolo - Ολοκληρωμένα Πειράματα, Τροφοδότηση και Τοποθέτηση Εικονικών Μηχανών										
Αρ.Πειράματος	Παράμετροι				1η Φάση – Τροφοδότησης Εικονικών Μηχανών		2η Φάση- Τοποθέτησης Εικονικών Μηχανών			
							1ος Τρόπος		2ος Τρόπος	
	m - Αριθμός Εφαρμο-γών	c – Κατηγο-ρίες VM	q - Αριθμός PM	u- Αριθμός vms	UGlobal	Χρόνος (minutes)	Active Pms	Χρόνος (minutes)	Active Pms	Χρόνος(minutes)
1	5	2	10	9	-683	0:00.0016	4	0:01.51	4	0:00.035
2	5	2	12	7	-52	0:00.0013	3	0:00.193	3	0:00.033
3	5	3	12	18	-1070	0:00.0033	9	09:15.325 (-)	9	0:00.223
4	5	3	16	14	-360	0:00.004	4	4:33.123	4	0:08.18
5	6	2	16	18	-504	0:00.00667	8	4:99:14 (-)	8	0:16.35
6	6	2	18	14	-211	0:00.167	5	4:02:16 (-)	5	0:20.81
7	6	3	18	25	-1837	0:00.03	7	9:55:66 (*)	6	9:43:51 (-)
8	6	3	22	15	-468	0:00.02	5	2:46:12 (-)	5	1:12:52 (-)
9	7	2	18	19	-364	0:00.0033	6	2:12:13 (-)	6	2:00.032 (-)
10	7	2	15	19	-180	0:00.005	6	1:32.28 (-)	6	0:16.63
11	7	3	18	22	-180	0:00.00333	11	2:12.33 (-)	11	1:35.201 (-)
12	7	3	15	15	-824	0:00.08	4	3:45.23 (-)	4	0:07.86
13	8	2	15	21	-232	0:00.0067	7	4:12.52 (-)	7	0:01.09
14	8	2	18	24	-1912	0:00.03	14	4:42.66 (-)	14	0:71.12 (-)
15	8	3	15	21	-808	0:00.0167	6	9:33:29 (*)	5	0:42.47(
16	8	3	18	28	-914	0:00.012	9	10:04.005 (-)	9	10.20.35 (-)
17	9	2	15	16	-134	0:00.01	4	2:38.18	4	0:02.45
18	9	2	18	14	-248	0:00.02	4	4:22.12 (-)	4	0:03.99

19	9	3	15	13	-48	0:00.057	3	0:02.76	3	0:00.495
20	9	3	18	21	-1212	0:00.01	11	1:34.12 (-)	11	0:01.55
21	10	2	18	19	-130	0:00.04	11	9:33.39 (*)	10	1:33.84
22	10	2	20	22	-828	0:00.01	7	9:02.33 (*)	6	3:56.49
23	10	3	18	18	-134	0:00.19	7	9:58.37 (*)	6	0:07.63
24	10	3	15	14	-360	0:00.05	3	0:17.45	3	0:01.35
25	10	4	15	22	-656	0:00.02	11	9:36.27 (*)	10	1:69.06
26	10	4	18	28	-1124	0:00.04	10	5:21.36 (-)	10	2:33.12 (-)
27	11	2	15	21	-412	0:00.003	7	9:12.33 (*)	6	0:18.41
28	11	2	18	13	740	0:00.003	3	0:04.72	3	0:00.44
29	11	3	18	18	-252	0:00.007	8	10:02.003 (*)	8	9:52:42 (*)
30	11	3	15	27	-1126	0:00.82	12	9:12.33 (*)	11	2:02.33 (-)
31	11	4	18	23	-324	0:00.013	9	9:57.12 (*)	9	10:02.39 (*)
32	11	4	22	19	-832	0:00.01	8	10:00.099 (*)	8	10:12.28 (*)
33	12	2	20	19	226	0:00.01	7	2:03.12 (-)	7	0:06.38
34	12	2	18	24	300	0:00.01	10	10:00.08 (*)	10	9:55.33 (*)
35	12	3	17	19	-66	0:00.02	8	9:59.46 (*)	7	0:79.18
36	12	3	20	21	-332	0:00.10	8	3:12.52 (-)	8	2:45.33 (-)
37	12	4	18	19	-38	0:00.04	7	10.46.32 (*)	7	10:02*43 (*)
38	12	4	15	24	-616	0:00.11	12	10.56.24 (*)	11	1:28.32 (-)

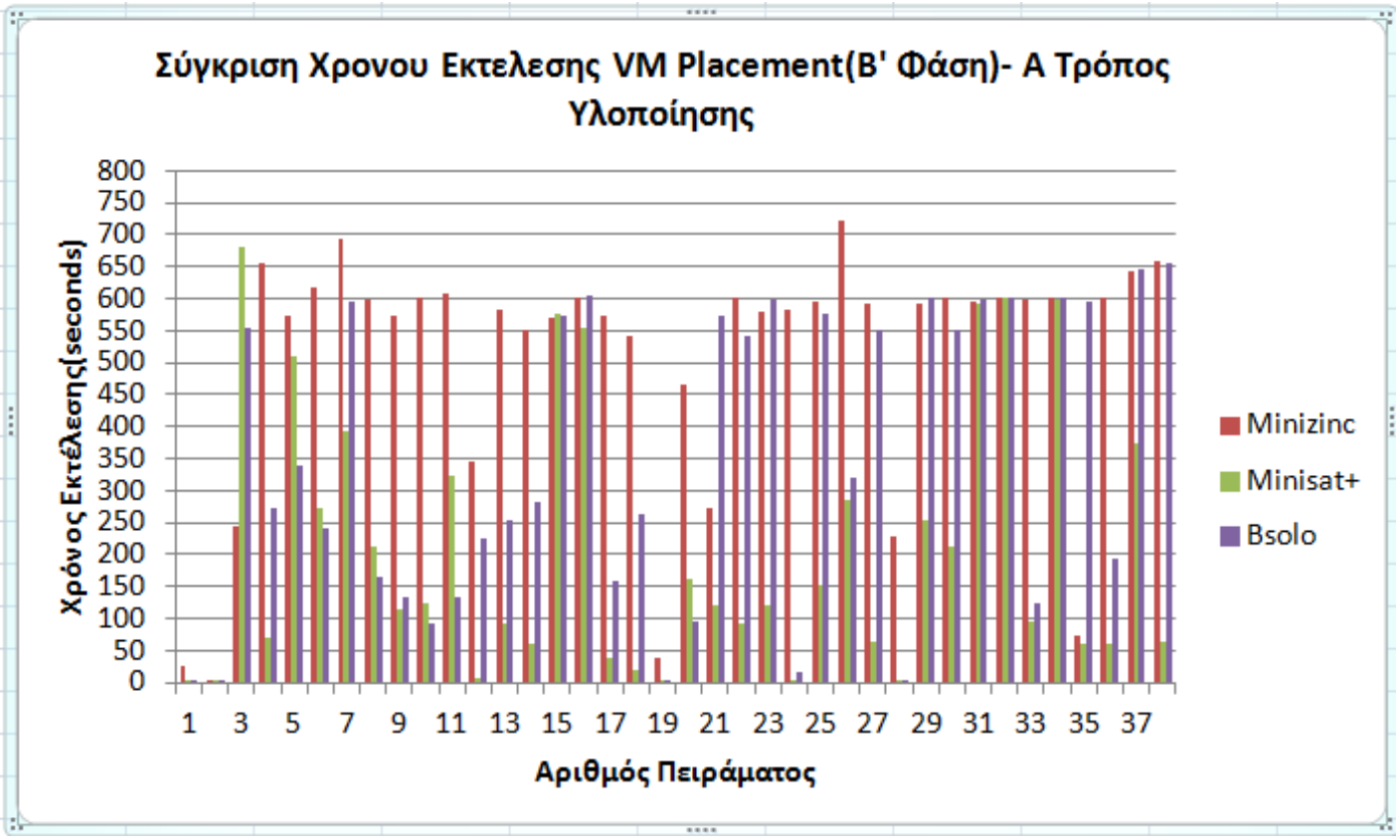
- Γραφική Αναπαράσταση - Σύγκριση Χρόνου Εκτέλεσης σε Ολοκληρωμένα Προβλήματα

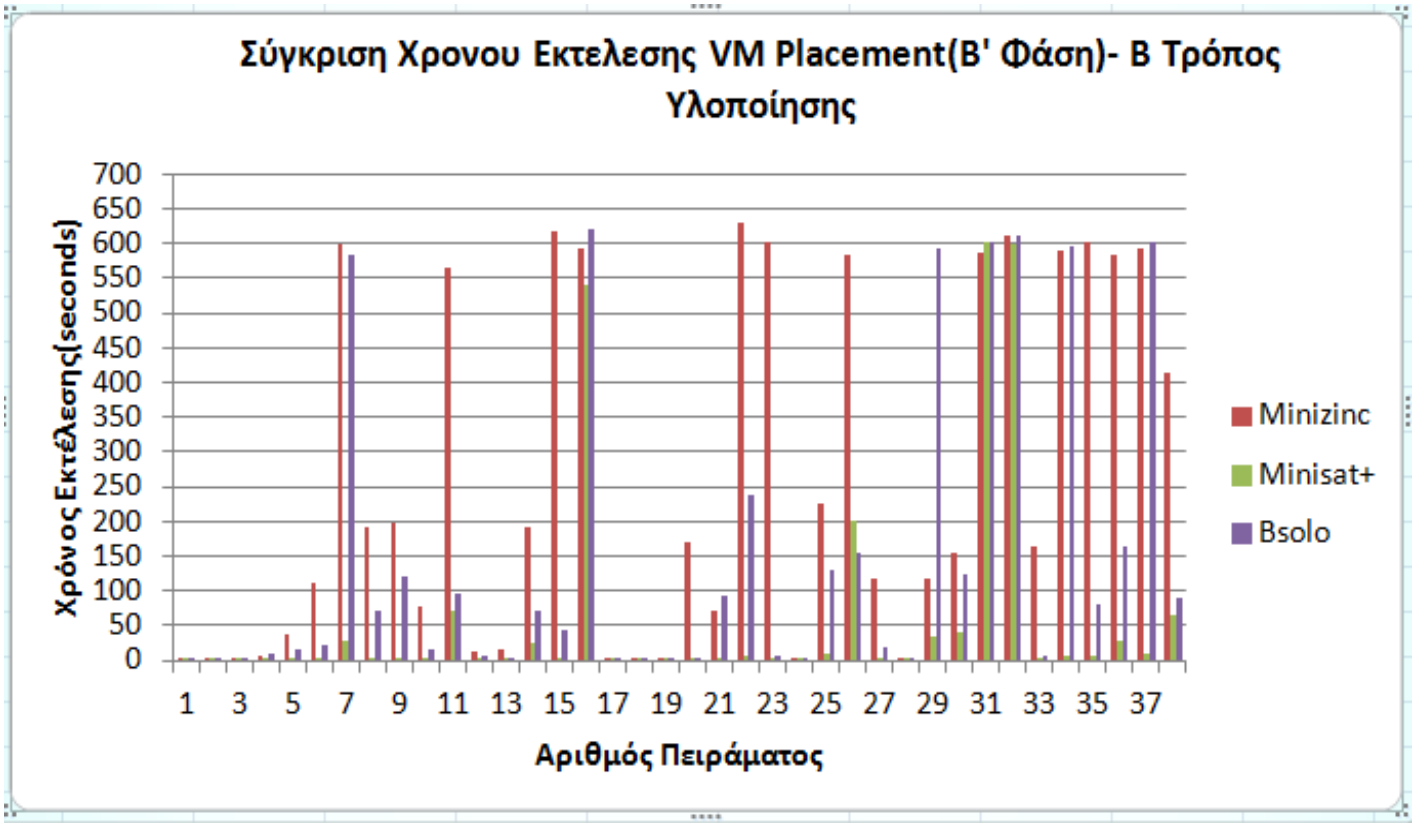
    1. Σύγκριση Χρόνου Εκτέλεσης για την φάση Τροφοδότησης Εικονικών Μηχανών (vm provisioning)



Η πιο πάνω γραφική παράσταση παρουσιάζει τους χρόνους εκτέλεσης της φάσης Τροφοδότησης Εικονικών Μηχανών για κάθε ένα από τους τρεις επιλυτές, για τα ολοκληρωμένα πειράματα που πραγματοποιήθηκαν. Όπως παρατηρούμε ο χρόνος εκτέλεσης του Bsolo είναι ο συντομότερος. Ο χρόνος εκτέλεσης του επιλυτή Minizinc είναι ο πιο αργός. Η διαφορά στους χρόνους εκτέλεσης ανάμεσα στους τρεις επιλυτές είναι αρκετά μεγάλη. Συγκεκριμένα, βλέπουμε ότι στους ψευδοδυαδικούς επιλυτές, Minisat+ και Bsolo ο χρόνος εκτέλεσης είναι πολύ λιγότερος σε σχέση με τον επιλυτή Minizinc.

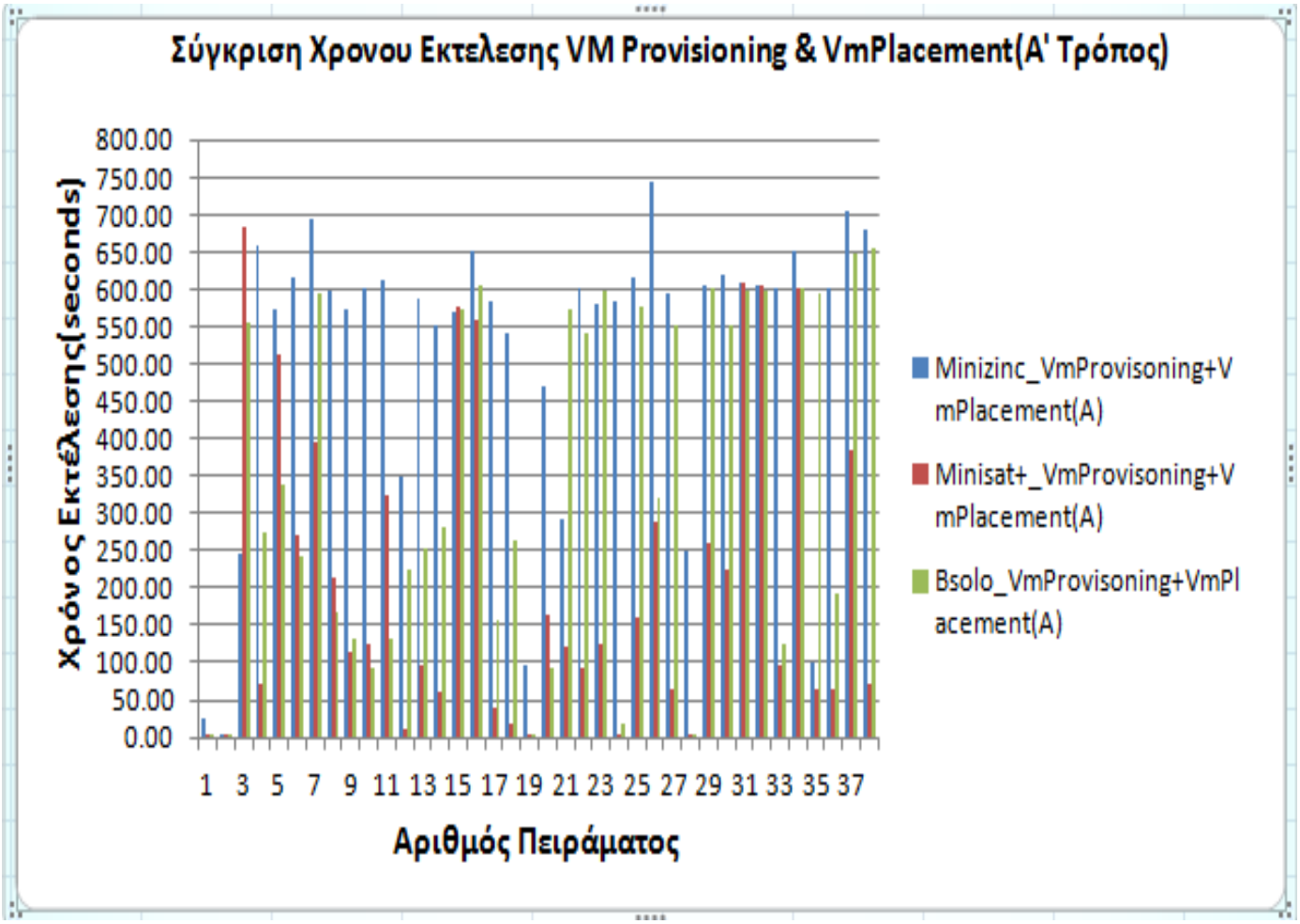
2. Σύγκριση Χρόνου Εκτέλεσης για την φάση Τοποθέτησης Εικονικών Μηχανών (vm placement) – Α & Β Τρόπος Υλοποίησης

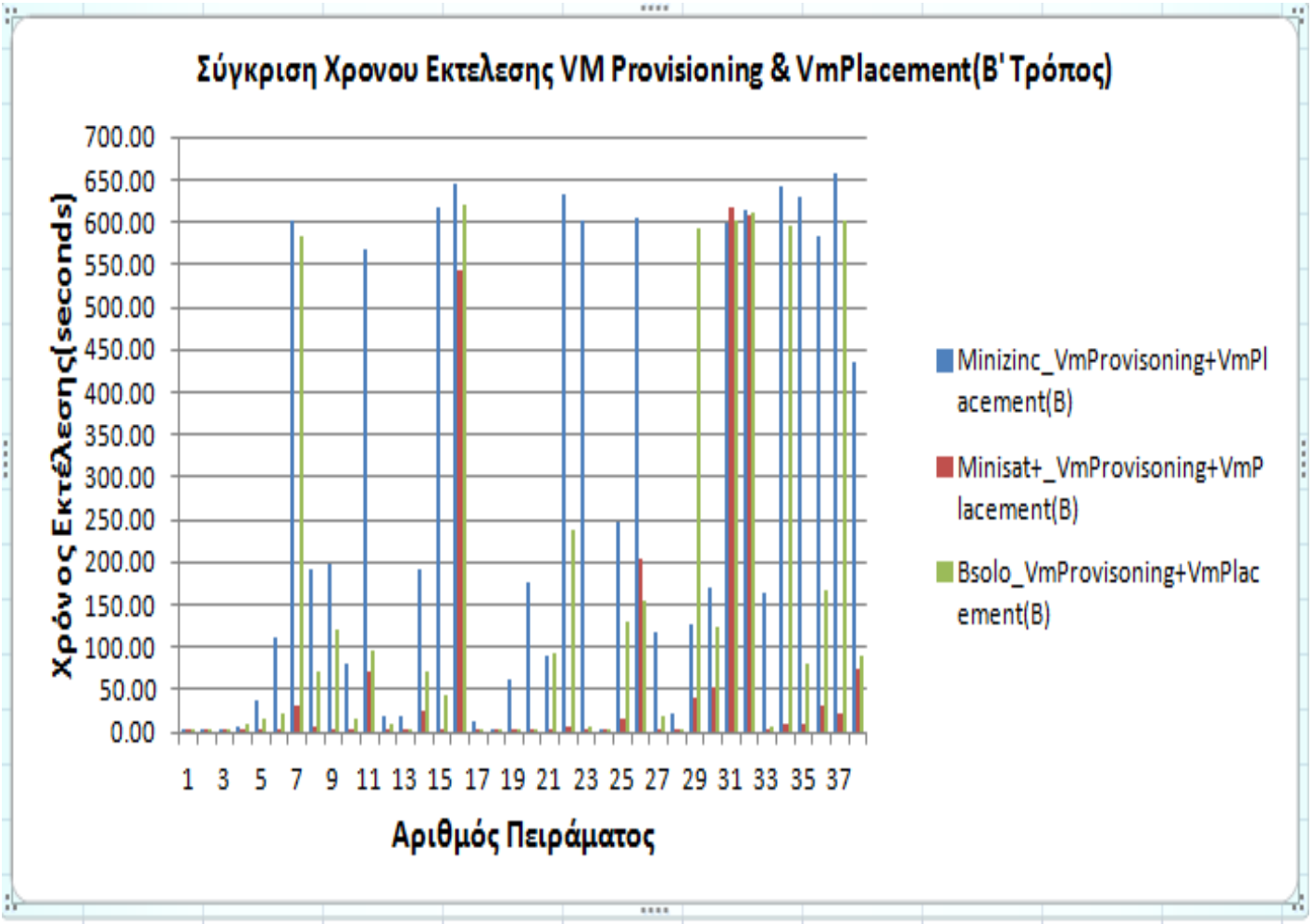




Οι δύο πιο πάνω γραφικές παραστάσεις παρουσιάζουν τους χρόνους εκτέλεσης της φάσης Τοποθέτησης Εικονικών Μηχανών για τον πρώτο(πάνω) και δεύτερο(κάτω) τρόπο υλοποίησης, για κάθε ένα από τους τρεις επιλυτές για τα ολοκληρωμένα πειράματα που πραγματοποιήθηκαν. Όπως παρατηρούμε οι χρόνοι εκτέλεσης στον δεύτερο τρόπο υλοποίησης της φάσης αυτής και για τους 3 επιλυτές, είναι μικρότεροι σε σχέση με τον πρώτο τρόπο υλοποίησης. Είναι προφανές ότι πάλι ο Minizinc είναι ο πιο αργός και στους δυο τρόπους υλοποίησης, σε σχέση με τους άλλους δυο επιλυτές. Σημαντικό είναι το γεγονός ότι τα αποτελέσματα στον χρόνο εκτέλεσης του Minisat+ με τον δεύτερο τρόπο υλοποίησης είναι πολύ ικανοποιητικά γιατί όπως βλέπουμε είναι ο πιο γρήγορος σε σχέση με τους άλλους δύο επιλυτές .

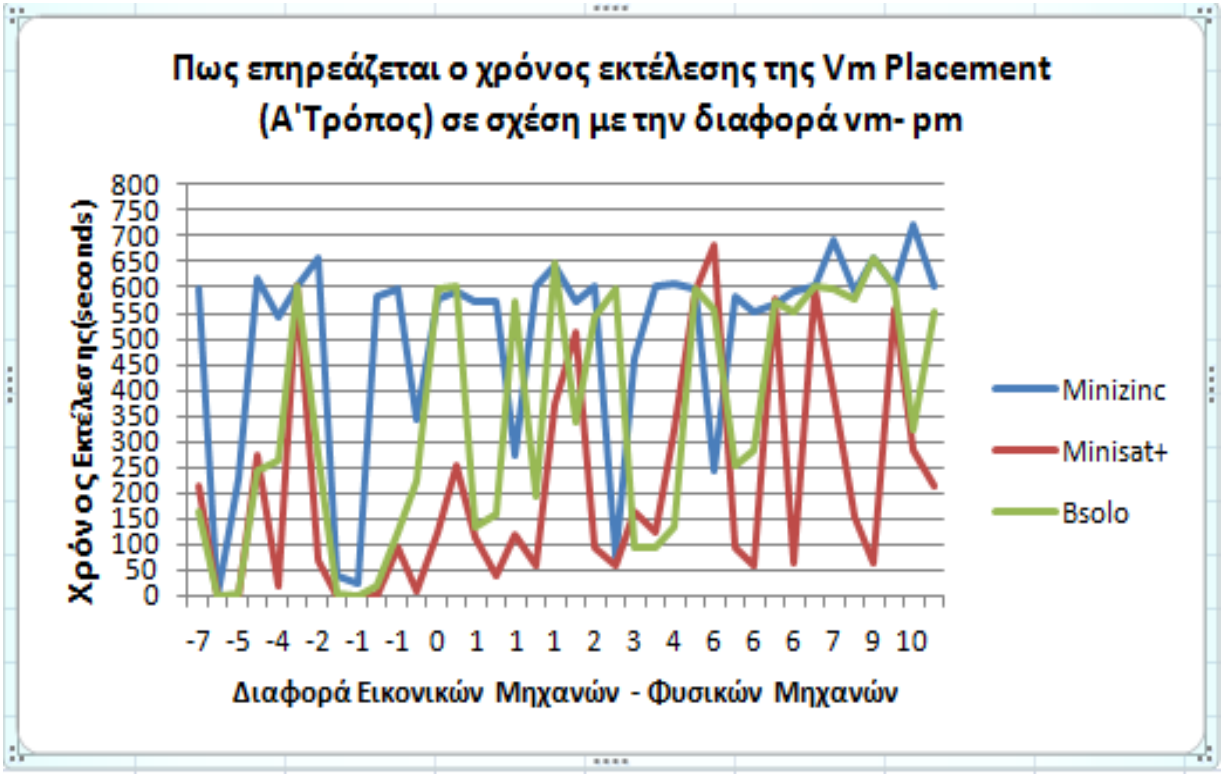
3. Σύγκριση Χρόνου Εκτέλεσης για Τροφοδότηση και Τοποθέτηση Εικονικών Μηχανών (Vm Provisioning + Vm Placement) – A & B Τρόπος

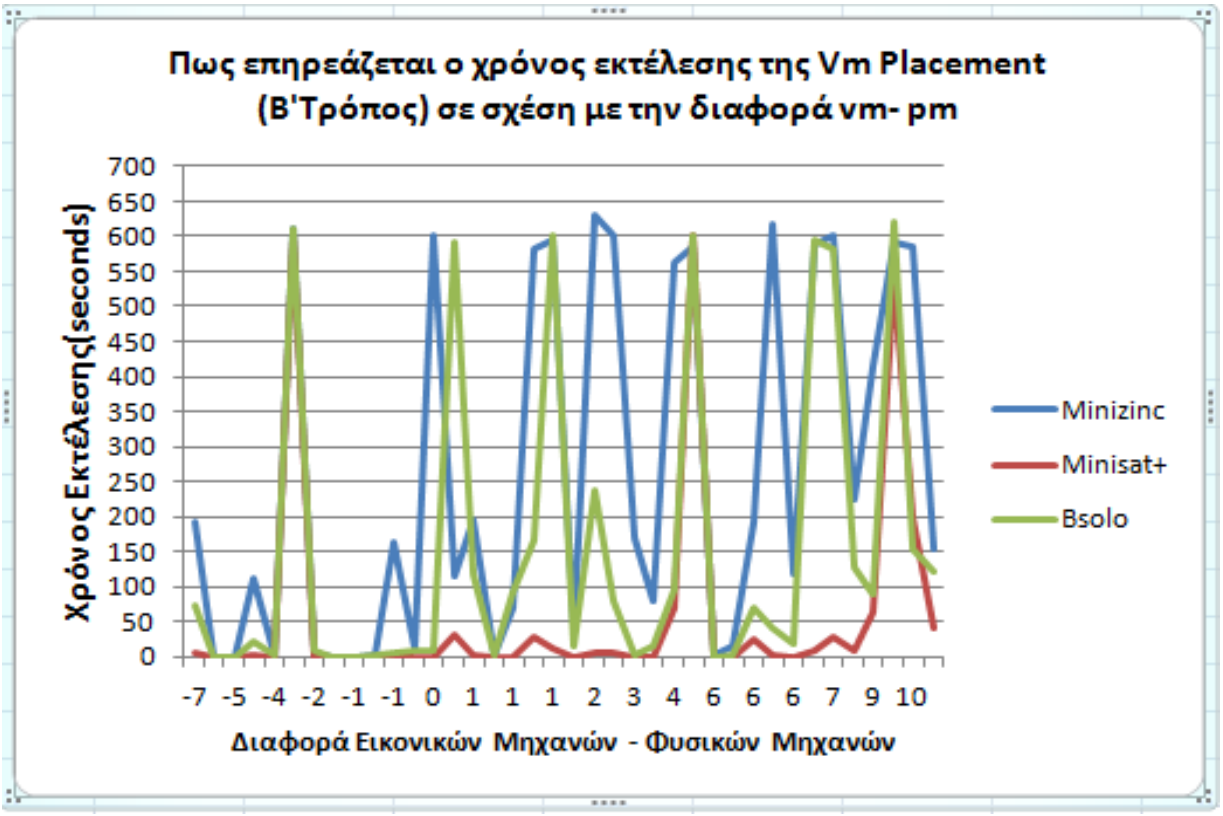




Οι δυο πιο πάνω γραφικές παραστάσεις, παρουσιάζουν τους ολοκληρωμένους χρόνους εκτέλεσης για τις δύο φάσεις τροφοδότησης και τοποθέτησης εικονικών μηχανών, για τους δυο τρόπους υλοποίησης της φάσης τοποθέτησης. Πάλι, όπως έχει παρατηρηθεί και στις προηγούμενες γραφικές παραστάσεις οι ολοκληρωμένοι χρόνοι εκτέλεσης των δύο φάσεων με τον δεύτερο τρόπο υλοποίησης της φάσης τοποθέτησης και για τους 3 επιλυτές, είναι μικρότεροι σε σχέση με τον χρόνο εκτέλεσης των δύο φάσεων με τον πρώτο τρόπο υλοποίησης της φάσης τοποθέτησης. Είναι προφανές ότι και πάλι ο επιλυτής Minizinc είναι ο πιο αργός και στους δυο τρόπους υλοποίησης σε σχέση με τους άλλους δυο επιλυτές. Σημαντικό είναι το γεγονός ότι τα αποτελέσματα στον χρόνο εκτέλεσης του Minisat+ με τον δεύτερο τρόπο υλοποίησης είναι πολύ ικανοποιητικά γιατί όπως βλέπουμε είναι ο πιο γρήγορος σε σχέση με τους άλλους δύο επιλυτές.

4. Γραφική Παράσταση για το πώς επηρεάζεται ο χρόνος εκτέλεσης της φάσης Τοποθέτησης Εικονικών Μηχανών (Α & Β Τρόποι Υλοποίησης) σε σχέση με την διαφορά εικονικών μηχανών και φυσικών μηχανών.





Στις δύο πιο πάνω γραφικές απεικονίζεται η επίδραση στον χρόνο εκτέλεσης της φάσης τοποθέτησης εικονικών μηχανών και με τους δύο τρόπους υλοποίησης από την διαφορά ανάμεσα στις φυσικές και εικονικές μηχανές. Όσο μεγαλώνει η διαφορά αυτή, είναι λογικό ότι τα προβλήματα γίνονται δυσκολότερα αφού αναζητούμε την ανάθεση εικονικών μηχανών στις φυσικές μηχανές με απώτερο σκοπό την ελαχιστοποίηση του αριθμού των ενεργών φυσικών μηχανών. Με βάση τις δυο πιο πάνω γραφικές, παρατηρούμε ότι οι χρόνοι εκτέλεσης και με τους δύο τρόπους υλοποίησης της φάσης τοποθέτησης επηρεάζονται από την διαφορά μεταξύ εικονικών και φυσικών μηχανών. Την μεγαλύτερη επίδραση στον χρόνο εκτέλεσης την έχουν οι επιλυτές Minizinc και Minisat+. Συγκεκριμένα ο επιλυτής Minizinc επηρεάζεται στον δεύτερο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών, αφού όπως βλέπουμε από την γραφική παράσταση, για μικρότερες τιμές στην διαφορά εικονικών και φυσικών μηχανών χρειάζεται λιγότερο χρόνο εκτέλεσης σε σχέση με τον χρόνο εκτέλεσης για μεγαλύτερες τιμές στην διάφορα εικονικών και φυσικών μηχανών όπου ο χρόνος εκτέλεσης του Minizinc είναι πολύ μεγαλύτερος. Όσων αφορά τον επιλυτή Minisat+, παρατηρούμε ότι όσο μεγαλώνει η διαφορά ανάμεσα στις εικονικές και φυσικές μηχανές, επηρεάζεται ο χρόνος εκτέλεσης του, ιδιαιτέρα με τον πρώτο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών αφού όπως βλέπουμε στην γραφική του επιλυτή αυτού έχει μια σταδιακή αύξηση στον χρόνο εκτέλεσης σε σχέση με την διαφορά μεταξύ εικονικών και φυσικών μηχανών.

7.2 Συμπεράσματα

7.2.1 Συμπεράσματα από Μέρος Α - Πειράματα για τη φάση Τροφοδότησης Εικονικών Μηχανών(VM Provisioning)

Με βάση τα πειράματα που έχουν πραγματοποιηθεί στο Μέρος Α της πειραματικής αξιολόγησης και με βάση τις γραφικές αναπαραστάσεις που προέκυψαν συμπεραίνουμε τα πιο κάτω:

- Με βάση το χρόνο εκτέλεσης της φάσης τροφοδότησης εικονικών μηχανών σε σχέση τις κατηγορίες εικονικών μηχανών βλέπουμε ότι η αύξηση των κατηγοριών εικονικών μηχανών(c), πολλές φορές επηρεάζει τον χρόνο εκτέλεσης της φάσης τροφοδότησης στον επιλυτή Minizinc. Παρατηρούμε διακυμάνσεις στο χρόνο εκτέλεσης στον επιλυτή Minizinc αφού μερικές φορές αυξάνεται ραγδαία καθώς αυξάνονται οι κατηγορίες εικονικών μηχανών, αλλά και μερικές φορές η αύξηση στον χρόνο δεν

είναι τόσο εμφανής. Για τους άλλους δυο επιλυτές Minisat+ και Bsolo είναι εμφανές ότι η αύξηση των κατηγοριών εικονικών μηχανών δεν επηρεάζει σχεδόν καθόλου τον χρόνο εκτέλεσης τους, μόνο στο Minisat+ βλέπουμε μια ελάχιστη αύξηση στα τελευταία πειράματα.

- Με βάση το χρόνο εκτέλεσης της φάσης τροφοδότησης εικονικών μηχανών σε σχέση με τον συντελεστή κόστους βλέπουμε ότι η αύξηση του συντελεστή κόστους(e), λίγες φορές επηρεάζει τον χρόνο εκτέλεσης της φάσης τροφοδότησης στον επιλυτή Minizinc. Παρατηρούμε διακυμάνσεις στο χρόνο εκτέλεσης στον επιλυτή Minizinc αφού μερικές φορές αυξάνεται πολύ αλλά και μερικές φορές η αύξηση στον χρόνο δεν είναι τόσο εμφανής. Συγκεκριμένα, βλέπουμε ότι η αύξηση στον χρόνο εκτέλεσης του Minizinc γίνεται στα σημεία όπου ο συντελεστής κόστους δεν είναι μεγάλος αριθμός. Ακόμη βλέπουμε ότι για μεγαλύτερους αριθμούς στον συντελεστή κόστους, ο χρόνος εκτέλεσης στον Minizinc μειώνεται αρκετά και γίνεται έως και πολύ μικρός. Για τους άλλους δυο επιλυτές, Minisat+ και Bsolo είναι εμφανές ότι η αύξηση του συντελεστή κόστους δεν επηρεάζει σχεδόν καθόλου το χρόνο εκτέλεσης τους. Σημαντικό είναι το γεγονός ότι στις περιπτώσεις όπου ο συντελεστής κόστους είναι μεγάλος αριθμός (7,8,9), ο Minisat+ είναι λίγο πιο αργός σε σχέση με τον Minizinc.

- Με βάση το χρόνο εκτέλεσης της φάσης τροφοδότησης εικονικών μηχανών σε σχέση με το άθροισμα των βαρυτήτων των εφαρμογών, βλέπουμε ότι η αύξηση του αθροίσματος των βαρυτήτων των εφαρμογών κάθε πειράματος, πολλές φορές επηρεάζει το χρόνο εκτέλεσης της φάσης τροφοδότησης στον επιλυτή Minizinc. Παρατηρούμε πολλές διακυμάνσεις στο χρόνο εκτέλεσης της φάσης τροφοδότησης των εικονικών μηχανών στον επιλυτή Minizinc αφού αρκετές φορές αυξάνεται πολύ. Συγκεκριμένα, βλέπουμε ότι η μεγαλύτερη αύξηση στον χρόνο εκτέλεσης του Minizinc γίνεται στα αρχικά σημεία όπου το συνολικό άθροισμα των βαρυτήτων δεν είναι μεγάλος αριθμός. Ακόμη βλέπουμε ότι καθώς αυξάνεται το συνολικό άθροισμα των βαρυτήτων η αύξηση στον χρόνο εκτέλεσης του Minizinc, τις περισσότερες φορές μειώνεται συγκεκριμένα, μειώνεται ο ρυθμός αύξησης του χρόνου εκτέλεσης. Για τους άλλους δυο επιλυτές Minisat+ και Bsolo είναι εμφανές ότι η αύξηση συνολικού αθροίσματος των βαρυτήτων δεν επηρεάζει σχεδόν καθόλου τον χρόνο εκτέλεσης τους. Σημαντικό είναι το γεγονός ότι στις περιπτώσεις όπου το άθροισμα των βαρυτήτων είναι μεγάλος αριθμός, ο Minisat+ είναι λίγο πιο αργός σε σχέση με

τον Minizinc. Φυσικά στις περιπτώσεις αυτές η διαφορά του χρόνου στους δύο αυτούς solver είναι ελάχιστη.

- Έτσι, ως επακόλουθο των τριών πιο πάνω συμπερασμάτων και ως γενικό συμπέρασμα για το πώς επηρεάζεται ο χρόνος εκτέλεσης της φάσης τροφοδότησης σε σχέση με την αύξηση των κατηγοριών εικονικών μηχανών, το συντελεστή κόστους και το άθροισμα των βαρυτήτων των εφαρμογών, παρατηρούμε ότι πολλές φορές όσο αυξάνεται ο αριθμός των κατηγοριών εικονικών μηχανών τόσο αυξάνεται και ο χρόνος εκτέλεσης του επιλυτή Minizinc. Επίσης, μερικές φορές όσο αυξάνεται ο αριθμός του συντελεστή κόστους τόσο πιο γρήγορος είναι ο Minizinc για την εύρεση λύσης. Σημαντικό είναι το γεγονός ότι ο συνδυασμός μεταξύ μικρού αριθμού για συντελεστή κόστους και μικρούς αριθμούς για τις βαρύτητες, τόσο πιο γρήγορα τερματίζει ο Minizinc.

- Όσων αφορά τον επιλυτή Minisat+, συμπεραίνουμε ότι για μεγάλο συντελεστή κόστους και για μεγάλους αριθμούς στις βαρύτητες των εφαρμογών, πολλές φορές χρειάζεται περισσότερο χρόνο εύρεσης της βέλτιστης λύσης της φάσης τροφοδότησης των εικονικών μηχανών σε σχέση με τον επιλυτή Minizinc.

- Όσων αφορά τον επιλυτή Bsolo, όπως έχουμε παρατηρήσει είναι ο πιο γρήγορος για την εύρεση λύσης στην φάση τροφοδότησης εικονικών μηχανών. Συγκεκριμένα, οι χρόνοι εκτέλεσης στον Bsolo είναι οι πιο μικροί και τίποτα από αυτά, κατηγορίες εικονικών μηχανών, συντελεστή κόστους και άθροισμα βαρυτήτων δεν επηρεάζει τον χρόνο εκτέλεσης του επιλυτή αυτού.

- Τέλος, με βάση την γραφική παράσταση για το πώς επηρεάζεται η τιμή του UGlobal σε σχέση με τις κατηγορίες εικονικών μηχανών, τον συντελεστή κόστους και το άθροισμα των βαρυτήτων, είναι εμφανές ότι η αύξηση των βαρυτήτων άλλα και του συντελεστή κόστους επηρεάζουν την αύξηση της συνάρτησης ικανοποίησης των εφαρμογών. Συγκεκριμένα, βλέπουμε ότι καθώς ο συντελεστής κόστους αυξάνεται, αυξάνεται και η τιμή της συνάρτησης αυτής. Επίσης στη γραφική η οποία απεικονίζει τις βαρύτητες των εφαρμογών βλέπουμε πολλές διακυμάνσεις εντούτοις είναι προφανές ότι καθώς αυξάνεται το άθροισμα των βαρυτήτων αυξάνεται και η τιμή του UGlobal. Σημαντικό είναι το γεγονός ότι η αύξηση των κατηγοριών εικονικών

μηχανών δεν επηρεάζει σχεδόν καθόλου την τιμή του UGlobal. Έτσι συμπεραίνουμε ότι στην μεγιστοποίηση του UGlobal καθοριστικός είναι ο ρόλος του συντελεστή κόστους και των βαρυτήτων των εφαρμογών.

7.2.2 Συμπεράσματα από Μέρος Β - Ολοκληρωμένα Πειράματα για τις φάσεις, Τροφοδότησης Εικονικών Μηχανών(VM Provisioning) και Τοποθέτησης Εικονικών Μηχανών στις Φυσικές Μηχανές(VM Placement)

- Με βάση τις γραφικές παραστάσεις που έχουν γίνει στο δεύτερο μέρος της πειραματικής ανάλυσης για τα ολοκληρωμένα περάματα των φάσεων τροφοδότησης και τοποθέτησης εικονικών μηχανών, συμπεραίνουμε ότι ο πρώτος τρόπος υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών ο οποίος παίρνει ως είσοδο από την φάση τροφοδότησης ένα αριθμό ο οποίος προσδιορίζει ποιος είναι ο συνολικός αριθμός εικονικών μηχανών που απαιτούνται για κάθε πρόβλημα είναι και για τους τρεις επιλυτές Minizinc, Minisat+ και Bsolo περισσότερο αργός σε σχέση με τον δεύτερο τρόπο υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών ο οποίος παίρνει ως είσοδο από την φάση τροφοδότησης εικονικών μηχανών ένα διάνυσμα το οποίο αναφέρεται σε κάθε κατηγορία εικονικής μηχανής και συγκεκριμένα παίρνει ένα αριθμό για κάθε κατηγορία εικονικών μηχανών, δηλαδή πόσες εικονικές μηχανές απαιτούνται για κάθε κατηγορία.

- Ως προς την ποιότητα της βέλτιστης λύσης της φάσης τοποθέτησης, δηλαδή πόσες φυσικές μηχανές είναι ενεργές σε σχέση με την ανάθεση των εικονικών μηχανών σε αυτές, συμπεραίνουμε ότι ο δεύτερος τρόπος υλοποίησης της φάσης αυτής έχει καλύτερα αποτελέσματα. Συγκεκριμένα, όπως αναφέραμε και προηγουμένως κάποια παραδείγματα τερματίζονταν στα 10 λεπτά. Έτσι και στους τρεις επιλυτές στα παραδείγματα αυτά παρατηρούμε ότι στον χρόνο των 10 λεπτών στον δεύτερο τρόπο υλοποίησης ο αριθμός των ενεργών φυσικών μηχανών είναι μικρότερος σε σχέση με τον πρώτο τρόπο υλοποίησης στον χρόνο των 10 λεπτών όπου ο αριθμός των φυσικών μηχανών πολλές φορές ήταν μεγαλύτερος.

- Επακόλουθο του πιο πάνω συμπεράσματος είναι το γεγονός ότι σε πολλά παραδείγματα τα όποια αργούσαν να τερματίσουν, στον πίνακα των πειραμάτων αναγράφεται ο χρόνος ο όποιος χρειαζόταν κάποιος επιλυτής για να εντοπίσει την

βέλτιστη λύση. Όπως παρατηρούμε και στους τρεις επιλυτές ο δεύτερος τρόπος υλοποίησης της φάσης τοποθέτησης, στα περισσότερα πειράματα σε μικρότερο χρόνο εκτέλεσης εντοπίζει την βέλτιστη λύση σε σχέση με τον πρώτο τρόπο υλοποίησης ο οποίος πολλές φορές δεν τερματίζει και σταματά σε προηγούμενες λύσεις.

- Τέλος, παρατηρούμε ότι ο δεύτερος τρόπος υλοποίησης της φάσης τοποθέτησης εικονικών μηχανών είναι ο γρηγορότερος και για τους τρεις επιλυτές και βγάζει και τις καλύτερες βέλτιστες λύσεις σε λιγότερο χρόνο. Σημαντικό είναι το γεγονός ότι ο επιλυτής Minisat+ με τον δεύτερο τρόπο υλοποίησης της φάσης τοποθέτησης πάντοτε τερματίζει (εκτός από 2-3 περιπτώσεις), ακόμα και σε πολύ δύσκολα προβλήματα όπου η διαφορά μεταξύ εικονικών και φυσικών μηχανών είναι αρκετά μεγάλη. Ο πρώτος τρόπος υλοποίησης του Minisat+ και οι δύο τρόποι υλοποίησης της φάσης τοποθέτησης του Bsolo βρίσκουν πολλές φόρες την βέλτιστη λύση(τερματίζοντας ή μη) αλλά δυσκολεύονται να αποφασίσουν ότι όντος αυτή είναι η βέλτιστη λύση όπως επίσης, λίγες είναι οι φόρες όπου δεν βρίσκουν την βέλτιστη λύση και δεν τερματίζουν σταματώντας σε προηγούμενες λύσεις.

- Συνοψίζοντας τα πιο πάνω συμπεράσματα βλέπουμε ότι οι επιδόσεις στον χρόνο εκτέλεσης των ψευδοδυαδικών επιλυτών οι οποίοι επιλύουν ψευδοδυαδικούς περιορισμούς με πεδία τιμών στις μεταβλητές 0 ή 1 είναι καλύτερες σε σχέση με τον επιλυτή Minizinc ο οποίος επιλύει προβλήματα ικανοποίησης περιορισμών με πεδία τιμών στις μεταβλητές τους ακέραιους και πραγματικούς αριθμούς.

7.3 Σύνοψη

Τα προβλήματα κατανομής πόρων έχουν πολλές εφαρμογές σε πολλές επιχειρήσεις όπως και στην καθημερινή μας ζωή. Στον κόσμο του υπολογιστικού νέφους και της εικονοποίησης τα προβλήματα αυτά έχουν μεγάλο ποσοστό επίδρασης και εφαρμογής αφού ο τρόπος επίλυσης προβλημάτων ανάθεσης εικονικών μηχανών σε φυσικές μηχανές έχει επηρεάσει πολλούς ερευνητές. Με τα προβλήματα αυτά έχουν ασχοληθεί πολλοί ερευνητές λόγω της δυσκολίας, του ενδιαφέροντας και αναμφισβήτητα λόγω της πρακτικής σημασίας τους.

Τα προβλήματα κατανομής πόρων έχουν πολλές και διαφορετικές προσεγγίσεις όμως όλα έχουν την ίδια δομή. Αρχικά πρέπει να πραγματοποιηθεί η τροφοδότηση και κατανομή των πόρων στις εικονικές μηχανές μεγιστοποιώντας παράλληλα την συνάρτηση ικανοποίησης των εφαρμογών(UGlobal) και έπειτα να πραγματοποιηθεί η διαδικασία τοποθέτησης αυτών των εικονικών μηχανών σε φυσικές μηχανές με απώτερο σκοπό την ελαχιστοποίηση των ενεργών φυσικών μηχανών. Ο στόχος τοποθέτησης μπορεί να είναι είτε μεγιστοποίηση της χρήσης των διαθέσιμων πόρων ή μπορεί να περιλαμβάνει εξοικονόμηση ενέργειας με την έννοια ότι μπορεί να κλείσουν/απενεργοποιηθούν κάποιοι εξυπηρετητές. Έτσι λοιπόν, οι αυτόνομοι αλγόριθμοι τοποθέτησης εικονικών μηχανών είναι σχεδιασμένοι έχοντας κατά νου τους παραπάνω στόχους.

Η τεχνολογία της εικονοποίησης διακομιστή(server virtualization) είναι μια αποτελεσματική προσέγγιση για την βελτίωση της αποδοτικότητας των πόρων και της εξοικονόμησης ενέργειας, ενώ παράλληλα παρέχει και εγγυήσεις επίδοσης για τις εφαρμογές. Έτσι χρησιμοποιώντας την τεχνολογία αυτή, ο χρόνος εκτέλεσης των εφαρμογών εγκλείεται σε μια εικονική μηχανή και μία ή περισσότερες εικονικές μηχανές φιλοξενούνται σε μια φυσική μηχανή. Επίσης, η τεχνολογία της εικονοποίησης μπορεί να παρέχει απομόνωση επίδοσης μεταξύ εικονικών μηχανών που τοποθετούνται στην ίδια φυσική μηχανή και να αποφεύγεται η υποβάθμιση επίδοσης που προκαλείται από άπληστες ή κακόβουλες εφαρμογές. Εξάλλου, είναι χρήσιμο να βελτιωθεί η αξιοποίηση των πόρων και η εξοικονόμηση κόστους.

Όπως έχουμε παρατηρήσει μέσα από τα πειράματα και μέσα από τις γραφικές παραστάσεις που έχουν προκύψει, η μορφή δεδομένου εισόδου για την φάση τοποθέτησης εικονικών μηχανών επηρεάζει τον χρόνο εύρεσης και ολοκλήρωσης της βέλτιστης λύσης. Συγκεκριμένα, η υλοποίηση της φάσης τοποθέτησης εικονικών μηχανών σε φυσικές μηχανές με δεδομένο εισόδου ένα διάνυσμα το οποίο αναφέρεται σε κάθε κατηγορία εικονικών μηχανών δηλαδή έναν αριθμό για κάθε κατηγορία εικονικών μηχανών ο οποίος αναφέρει πόσες εικονικές μηχανές απαιτούνται για κάθε κατηγορία, είναι αποδοτικότερη από άποψη χρόνου και ποιότητα λύσης σε σχέση με την υλοποίηση της φάσης αυτής η οποία έχει δεδομένο εισόδου έναν αριθμό ο οποίος προσδιορίζει ποιος είναι ο συνολικός αριθμός εικονικών μηχανών που απαιτούνται.

Μέσα από τα δύο μέρη των πειραματικών αξιολογήσεων, προκύπτει το συμπέρασμα ότι ο επιλυτής Minizinc λειτουργεί χωρίς κάποια στρατηγική και σε κάποιες περιπτώσεις τα αποτελέσματα του δυνατόν να μην ήταν τα αναμενόμενα, σε σχέση με τους ψευδοδυαδικούς επιλυτές Minisat+ και Bsolo. Συγκεκριμένα, μέσα από τις γραφικές παραστάσεις οι οποίες αφορούν τους χρόνους εκτέλεσης των δυο φάσεων τροφοδότησης και τοποθέτησης εικονικών μηχανών, βλέπουμε πολλές διακυμάνσεις στον χρόνο εκτέλεσης του επιλυτή Minizinc σε σχέση με τους άλλους δύο επιλυτές.

Συνοψίζοντας, συμπεραίνουμε ότι ο προγραμματισμός με περιορισμούς για την τοποθέτηση εικονικών μηχανών μπορεί να μειώσει αποτελεσματικά τον αριθμό των φυσικών μηχανών για την επίτευξη του στόχου, της μείωσης της λειτουργικού κόστους, βελτίωση της χρήσης των πόρων και την μείωση του χρόνου εκτέλεσης. Συγκεκριμένα, βάση των πιο πάνω συμπερασμάτων, βλέπουμε ότι οι επιδόσεις στον χρόνο εκτέλεσης των ψευδοδυαδικών επιλυτών οι οποίοι επιλύουν ψευδοδυαδικούς περιορισμούς με πεδία τιμών στις μεταβλητές 0 ή 1 είναι καλύτερες σε σχέση με τον επιλυτή Minizinc ο οποίος επιλύει προβλήματα ικανοποίησης περιορισμών με πεδία τιμών στις μεταβλητές τους ακέραιους και πραγματικούς αριθμούς.

7.4 Μελλοντική Εργασία

Όπως έχουμε παρατηρήσει μέσα από τα πειράματα του δεύτερου μέρους οι τρεις επιλυτές βρίσκουν την βέλτιστη λύση σε διαφορετικούς χρόνους ή μερικοί από αυτούς δεν προσεγγίζουν καν την βέλτιστη λύση και δεν τερματίζουν, σταματώντας σε προηγούμενες λύσεις. Έτσι μια μέθοδος η οποία θα μπορούσε να εφαρμοστεί με σκοπό να βελτιστοποιηθούν οι χρόνοι εκτέλεσης είναι η υλοποίηση του προβλήματος και σε άλλους επιλυτές υποστήριξης προτασιακής ικανοποιησιμότητας(sat solver), ψευδοδυαδικούς επιλυτές ή σε επιλυτές ικανοποίησης περιορισμών με πεδία τιμών πραγματικούς ή ακέραιους αριθμούς όπως είναι ο Minizinc.

Παράλληλα, κάτι το οποίο θα μπορούσε να βελτιστοποιήσει την λύση του Minizinc θα ήταν να χρησιμοποιηθεί κάποιο είδους ευρετικό με διαφορετικά χαρακτηριστικά από αυτά που χρησιμοποιηθήκαν στις υλοποιήσεις του Minizinc με σκοπό να βοηθά στην εύρεση της λύσεις σε μικρότερο χρόνο.

Μια άλλη μέθοδος η οποία θα μπορούσε να βελτιστοποιήσει τις λύσεις και τον χρόνο εκτέλεσης είναι υλοποίηση της προσέγγισης που αναφέρθηκε στο κεφάλαιο 4, «Τροφοδότηση των Πόρων με περιορισμούς προϋπολογισμού για προσαρμοστικές εφαρμογές στο Υπολογιστικό Νέφος», με προγραμματισμό με περιορισμούς.[12]

Τέλος, όπως έχουμε δει στο κεφάλαιο 4 έχουν προταθεί πολλοί αλγόριθμοι οι οποίοι έχουν χρησιμοποιηθεί για να λυθεί το πρόβλημα της τοποθέτησης εικονικών μηχανών. Αφού λοιπόν στην παρούσα διπλωματική εργασία έχει υλοποιηθεί το πρόβλημα με τη μέθοδο ικανοποίησης περιορισμών θα μπορούσε να επεκταθεί με την υλοποίηση άλλων μεθόδων υλοποίησης του παρόντος προβλήματος.

Εν τέλει, σημαντικό είναι το γεγονός ότι το πρόβλημα της τοποθέτησης εικονικών μηχανών μπορεί εύκολα να επεκταθεί για τον λόγο ότι οι περιορισμοί του προβλήματος είναι περίπου οι ίδιοι κάθε φορά, συγκεκριμένα υπάρχει ένα σύνολο από εικονικές μηχανές και ένα σύνολο από φυσικές μηχανές. Τα δύο είδη μηχανών έχουν προκαθορισμένη ικανότητα σε πόρους και στόχος είναι να τοποθετηθούν οι εικονικές μηχανές σε όσο το δυνατό λιγότερες ενεργές φυσικές μηχανές. Έτσι, συμπεραίνουμε ότι το παρόν πρόβλημα της τοποθέτησης είναι επεκτάσιμο πρόβλημα αφού, οι περιορισμοί του προβλήματος είναι παρόμοιας φύσεως και το μόνο το οποίο θα αλλάζει θα είναι οι απαιτήσεις και η μορφή αναπαράστασης του προβλήματος.

Βιβλιογραφία

[1] Anjana Shankar (09305920), ”Virtual Machine Placement in Computing Clouds” Guided By: Prof.Umesh Bellur April 10, 2010.

[2] Fadi A. Aloul, "Search techniques for SAT-based Boolean optimization" The Franklin Institute. Published by Elsevier Ltd, 2006.

[3] Fadi A. Aloul, Arathi Ramani, Igor L. Markov, Karem A. Sakallah “A Backtrack-Search Pseudo-Boolean Solver and Optimizer”, Department of Electrical Engineering and Computer Science, University of Michigan, Submitted SAT ’2002.

[4] Federico Heras,Vasco Manquinho ,Joao Marques-Silva"On Applying Unit Propagation-Based Lower Bounds in Pseudo-Boolean Optimization", Association for the Advancement of Artificial Intelligence (www.aaai.org), 2008.

[5] Hien Nguyen Van, Fr´ed´eric Dang Tran Orange Labs, Jean-Marc Menaud Departement Informatique, ´ Ecole des Mines de Nantes “Autonomic virtual resource management for service hosting platforms”, published in Workshop on Software Engineering Challenges in Cloud Computing, Vancouver, Canada : Canada (2009).

[6] Introduction to Cloud Computing Architecture White Paper 1st Edition, June 2009 Sun Microsystems, Inc.

[7] Kim Marriott and Peter J. Stuckey, A MiniZinc Tutorial.

[8] Michael Armbrust et al.”A view of Cloud Computing”, Communications of the Acm, April 2010, vol.53,no.4.

[9] Niklas Een,Niklas Sorensson, "Translating Pseudo-Boolean Constraintsi nto SAT",Journal on Satisfiability, Boolean Modeling and Computation 2 (2006) 1-25.

[10] Olivier Bailleux,Yacine Boufkhad,Olivier Roussel, "A Translation of Pseudo-Boolean Constraints to SAT",Journal on Satisfiability, Boolean Modeling and Computation 2 (2006) 191-200.

[11] Olivier Roussel, Vasco M. Manquinho , "Pseudo Boolean and Cardinality Constraints",Handbook of Satisfiability Armin Biere,Marijn Heule,Hans van Maaren and Toby Walsh(Eds) IOS Press, 2009.

[12] Qian Zhu, Gagan Agrawai , “Resource Provisioning with Budget Constraints for Adaptive Applications in Cloud Environments” , Department of Computer Science and Engineering Ohio State University.

[13] Ronnie D.Caytiles, Sungunk Lee and Byungjoo Park, “Cloud Computing: The next Computing Paradigm”,Department Of Multimedia Engineering, Hannam University 133 ojeong-dong, International Journal of Multimedia and Ubiquitous Engineering Vol. 7, No. 2, April, 2012.

[14] Sara Angeles, “Virtualization vs. Cloud Computing: What's the Difference?” , BusinessNewsDaily Staff Writer | January 20, 2014.

[15] Yonghong Yu, Yang Gao , Nanjing University “ Constraint Programming –Based Virtual Machines Placement Algorithm in Datacenters”.