ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΤΟΧΑΣΤΙΚΟΙ ΕΥΡΙΣΤΙΚΟΙ ΜΗΧΑΝΙΣΜΟΙ ΤΟΠΙΚΗΣ ΑΝΑΖΗΤΗΣΗΣ ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΙΚΑΝΟΠΟΙΗΣΗΣ ΠΕΡΙΟΡΙΣΜΩΝ

ΚΑΡΠΟΥΖΗΣ ΑΘΑΝΑΣΙΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:
Dr. ΣΤΕΡΓΙΟΥ ΚΩΝΣΤΑΝΤΙΝΟΣ

Θεσσαλονίκη 2014

Αφιερώνεται
στον πολυαγαπημένο μου
παππού Θανάση...

Περίληψη

Τα προβλήματα ικανοποίησης περιορισμών και οι υποκλάσεις τους όπως είναι τα MAX-CSPs είναι θεμελιώδη προβήματα της Υπολογιστικής Θεωρίας και της Συνδιαστικής Βελτιστοποίησης. Αυτά τα προβλήματα στην γενική τους μορφή ανήκουν στην κλάση NP-complete και NP-hard αντίστοιχα όπως έχει δειχθεί και παρόλο που έχουν προταθεί συστηματικές μέθοδοι για την αντιμετώπισή τους, οι ευριστικοί μηχανισμοί τοπικής αναζήτησης έχουν αποδειχθεί ιδιαίτερα ανταγωνιστικοί. Στην παρούσα διπλωματική εργασία, αναλύσαμε, υλοποιήσαμε και συγκρίναμε την αποδοτικότητα μερικών από τους βασικότερους και ευρέως γνωστούς ευριστικούς μηχανισμούς όπως είναι των Ελάχιστων Συγκρούσεων (Min-Conflicts), του συνδιασμού του με την Τυχαία Περιήγηση (Min-Conflicts + Random Walk), της Αναζήτησης Ταμπού (Tabu Search), της Τοπικής Ακτινικής Αναζήτησης (Local Beam Search) και της Στοχαστικής Τοπικής Ακτινικής Αναζήτησης (Stochastic Local Beam Search). Ακόμη, προτείνουμε μια βελτιωμένη εκδοχή της Αναζήτησης Ταμπού την οποία αποκαλούμε Δυναμική Αναζήτηση Ταμπού που δύναται να μεταβάλλει δυναμικά το μέγεθος της λίστας ταμπού χρησιμοποιώντας τις πληροφορίες που συλλέγει κατά τη διάρκεια της αναζήτησης. Τα αποτελέσματα των πειραμάτων δείχνουν ότι η παραλλαγή της Αναζήτησης Ταμπού που υλοποιήσαμε πετυχαίνει καλύτερη απόδοση από την Αναζήτηση Ταμπού στην οποία το μέγεθος της λίστας ταμπού είναι προκαθορισμένο και ότι υπερέχει κατά πολύ έναντι των άλλων μεθόδων. Τα πειράματα διεξήχθησαν πάνω σε κλάσεις προβλημάτων δυαδικών περιορισμών.

Abstract

The constraint satisfaction problem and its subclasses like MAX-CSPs, are fundamental problems in computing theory and combinatorial optimization. These problems are proved to be *NP-complete* and *NP-hard* respectively in general and although systematic methods have been proposed to address them, incomplete approaches based on local search have been shown to be very promising in this field. In this study, we analysed and compared the effectiveness of some well known local search heuristic methods such as Min-Conflicts, Min-Conflicts combined with Random Walk, Tabu Search, Local Beam Search and Stochastic Local Beam Search. We also propose an improved version of Tabu Search which we call Dynamic Tabu Search that can dynamically update its tabu list size using information during search. Experimental results demonstrate that our new tabu search algorithm achieves better performance than a tabu algorithm with a fixed list size and dominates the other heuristics on many classes of binary CSPs and MAX-CSPs.

Ευχαριστίες

Στα πλαίσια αυτής της διπλωματικής εργασίας θα ήθελα να ευχαριστήσω τον κύριο Δρ. Κωνσταντίνο Στεργίου για την πολύτιμη και καθοριστική βοήθεια του η οποία συντέλεσε στην επιτυχή ολοκλήρωση της διπλωματικής εργασίας. Εν συνεχεία θα ήθελα να ευχαριστήσω τους γονείς μου και τον αδερφό μου για την συνεχή στήριξή τους καθώς και για την συμβολή τους στην πνευματική μου συγκρότηση. Ακόμη, θα ήθελα να ευχαριστήσω την συνοδοιπόρο μου Μυρσίνη για την βοήθεια και την εμψύχωση που μου προσέφερε σε δυσκολίες που συνάντησα. Τέλος δεν μπορώ να παραλείψω τους φίλους μου που ο καθένας με τον τρόπο του συνέδραμε ευεργετικά.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Μια ιδιαίτερη κλάση στο πεδίο της Τεχνητής Νοημοσύνης και γενικότερα της Επιστήμης Υπολογιστών αποτελούν τα Προβλήματα Ικανοποίησης Περιορισμών (Constraint Satisfaction Problems). Τα CSPs είναι θεμελιώδη για την επίλυση πολλών προβλημάτων στην Αυτοματοποιημένη Συλλογιστική, στην Μηχανική Όραση, στον σχεδιασμό VLSI, στην ρομποτική, στην σχεδίαση δικτύων υπολογιστών, στη θεωρία των γράφων καθώς και στην Επιχειρησιακή Έρευνα. Πολλά προβλήματα έχουν αναχθεί σε CSPs με μεγάλη επιτυχία όπως ο Χρωματισμός Γράφων, ο Χρονοπρογραμματισμός Διεργασιών, η Χρονοδρομολόγηση και γενικότερα προβλήματα ανάθεσης. Στόχος σ΄ αυτά τα προβλήματα είναι να βρεθούν λύσεις που να ικανοποιούν ένα δεδομένο σύνολο περιορισμών. Στην πολύ γενική τους μορφή μπορεί να θεωρηθεί ότι ανήκουν στην κλάση *NP-Complete* προβλημάτων και αυτό προκύπτει εύκολα από το γεγονός ότι γνωστά *NP-Complete* προβλήματα έχουν αναχθεί σε αυτήν τη μορφή.

Αρκετά δημοφιλείς για την αντιμετώπισή τους γίνονται ολοένα και περισσότερο οι ευριστικοί μηχανισμοί τοπικής αναζήτησης. Παρόλο που δεν είναι πλήρεις, δηλαδή δεν εγγυώνται ότι θα βρούν λύση σε κάποιο πρόβλημα, είναι ιδιαιτέρως αποδοτικοί σε σύγκριση με τις μεθόδους εξαντλητικής αναζήτησης του χώρου καταστάσεων των προβλημάτων. Το ιδιαίτερο χαρακτηριστικό τους

είναι ότι επεξεργάζονται μία κατάσταση ενός προβλήματος προσπαθώντας να τη βελτιώσουν σύμφωνα με μία συνάρτηση αποτίμησης κόστους και η πορεία της αναζήτησης έχει ώς σημείο έναρξης αυτήν την κατάσταση. Στην τοπική αναζήτηση συγκαταλέγεται μία πληθώρα ευριστικών μηχανισμών που έχουν ως στόχο να ελαχιστοποιήσουν το κόστος ενός προβλήματος. Στην παρούσα διπλωματική εργασία εξετάζονται και συγκρίνονται μερικοί από τους ισχυρότερους ευριστικούς μηχανισμούς όπως είναι των *Ελάχιστων Συγκρούσεων* (Min-Conflicts) και του συνδιασμού του με *Τυχαία Περιήγηση* (Min Conflicts + Random Walk), της *Αναζήτησης Ταμπού* (Tabu Search), της *Τοπικής Ακτινικής Αναζήτησης* (Local Beam Search) και της Στοχαστικής της εκδοχής (Stochastic Local Beam Search). Εκτός αυτών, έγινε η υλοποίηση μίας δικής μας παραλλαγής της Αναζήτησης Ταμπού την οποία καλούμε Δυναμική Αναζήτηση Ταμπού (Dynamic Tabu Search).

1.1 Μοντελοποίηση Προβλημάτων

Η ικανοποίηση περιορισμών κατά βάση αποτελείται από αναθέσεις τιμών σε μεταβλητές με τέτοιο τρόπο ώστε να πληρούνται ορισμένες απαιτήσεις (περιορισμοί). Η μοντελοποίηση των προβλημάτων με βάση αυτό το πρότυπο παρέχει ευελιξία στον τρόπο αντιμετώπισής τους αφού τα πεδία τιμών και οι περιορισμοί προσαρμόζονται εύκολα στα πλαίσια των εκάστοτε απαιτήσεων. Ένα απλό αλλά αντιπροσωπευτικό παράδειγμα αποτελεί η κατασκευή ωρολόγιου προγράμματος μαθημάτων στα Ακαδημαϊκά Ιδρύματα. Ως περιορισμοί μπορούν να οριστούν τα εξής: Κανένα μάθημα δεν πρέπει να επικαλύπτεται με κάποιο άλλο, οι ώρες διδασκαλίας κάθε ημέρας πρέπει να ανήκουν σ΄ ένα έυρος 12 ωρών, κάποιοι καθηγητές είναι συγκεκριμένες μέρες διαθέσιμοι κλπ. Ο σχεδιασμός ενός τέτοιου προγράμματος απαιτεί την κατά το μέγιστο δυνατή ικανοποίηση των απαιτήσεων.

1.2 Προβλήματα Ικανοποίησης Περιορισμών

Ορισμός 1.1: Ως Πρόβλημα Ικανοποίησης Περιορισμών [1] P ορίζεται μία πλειάδα (X, D, C) όπου $X = \{x_1, x_2, \ldots, x_n\}$ ένα πεπερασμένο πλήθος n μεταβλητών, $D = \{D(x_1), D(x_2), \ldots, D(x_n)\}$ ένα σύνολο από πεδία τιμών και $C = \{c_1, c_2, \ldots, c_m\}$ είναι ένα σύνολο m περιορισμών. Για κάθε μεταβλητή $x_i \in X$, το $D(x_i)$ είναι ένα πεπερασμένο πεδίο τιμών της. Κάθε περιορισμός $c_i \in C$ ορίζεται ως ένα ζεύγος $(var(c_i), rel(c_i))$, όπου $var(c_i) = \{x_{j_1}, \ldots, x_{j_k}\}$ είναι ένα ταξινομημένο υποσύνολο του X το οποίο καλείται πεδίο εφαρμογής (scope) του περιορισμού c_i και $rel(c_i)$ είναι υποσύνολο του Καρτεσιανού γινομένου $D(x_{j_1}) \times \cdots \times D(x_{j_k})$ το οποίο καθορίζει τους επιτρεπόμενους συνδιασμούς τιμών για τις μεταβλητές στο $var(c_i)$. Η απόδοση μίας τιμής σε μία μεταβλητή ορίζεται ως απλή ανάθεση, η απόδοση τιμών σε όλες της μεταβλητές ορίζεται ως πλήρης ανάθεση και η ανάθεση με την οποία δεν παραβιάζεται κανένας περιορισμός καλείται συνεπής. Λύση σε ένα πρόβλημα αποτελεί μία συνέπης πλήρης ανάθεση. Για λόγους απλούστευσης, στην παρούσα διατριβή, θα χρησιμοποιείται ο όρος «κατάσταση» για να υποδηλώσουμε μία πλήρη ανάθεση. Είναι προφανές ότι το πλήθος των πιθανών καταστάσεων ενός προβλήματος είναι το Καρτεσιανό Γινόμενο των πεδίων τιμών κάθε μεταβλητής *State Space* $= \prod_{i=1}^{n} D(x_i)$. Ένα CSP θεωρείται **συνεπές** αν έχει τουλάχιστον μία λύση, αλλιώς ονομάζεται **ασυνεπές**.

1.3 Γράφοι Περιορισμών

Κάθε CSP μπορεί να αναπαρασταθεί ως Γράφημα Περιορισμών (*Contstraint Graph*). Σε ένα Γράφημα Περιορισμών, κάθε τόξο ή ακμή αποτελεί έναν περιορισμό και οι κόμβοι αντιστοιχούν σε μεταβλητές του προβλήματος. Βαθμός κόμβου καλείται το πλήθος των ακμών που πρόσκεινται στον κόμβο. Εξ΄ ορισμού ένα τόξο συνδέει δύο μόνο κόμβους με αποτέλεσμα να μην είναι αυτού

του είδος ο γράφος επαρκής για την αναπαράσταση ν-αδικών περιορισμών παρά μόνο δυαδικών[1].

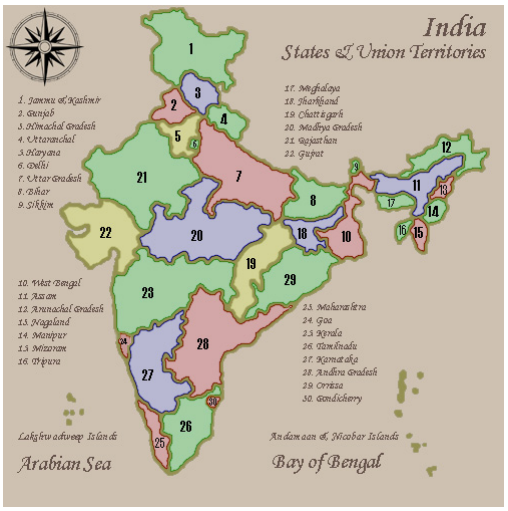
Εισάγεται λοιπόν η έννοια των Υπεργράφων Περιορισμών (*Contstraint Hypergraph*), για την γενίκευση της αναπαράστασης, στους οποίους κάθε κορυφή περιλαμβάνει ένα πλήθος από μεταβλητές που ανήκουν σε κάποιον περιορισμό c_i. Στην παρούσα εργασία δεν θα αναφερθούμε εκτενέστερα στα Υπεργραφήματα επειδή θα ασχοληθούμε κατά βάση με προβλήματα δυαδικών περιορισμών. Άλλωστε κάθε Πρόβλημα Ικανοποίησης Περιορισμών μπορεί να αναχθεί[2] σε πρόβλημα δυαδικών περιορισμών [2].

Πολλές φορές η δομή ενός γραφήματος περιορισμών μπορεί να χρησιμοποιηθεί για την απλοποίηση της διαδικασίας επίλυσης, παρέχοντας σε μερικές περιπτώσεις εκθετική μείωση της πολυπλοκότητας. Ένα αντιπροσωπευτικό παράδειγμα αποτελεί ο χρωματισμός γράφων όπως φαίνεται στο σχήμα 1.1

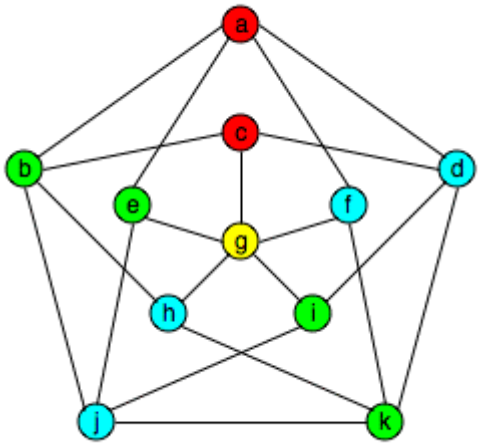
[1] Αν σε κάθε περιορισμό $c_i \in C$ υπάρχουν 2 μόνο μεταβλητές τότε έχουμε δυαδικούς περιορισμούς

[2] Μερικές φορές αυτό γίνεται με χρήση βοηθητικών μεταβλητών.

Σχήμα 1.1 (α′)Πρόβλημα Χρωματισμού Χάρτη: Για έναν δεδομένο αριθμό διαθέσιμων χρωμάτων, πρέπει κάθε πόλη (ή χώρα) να χρωματιστεί με ένα από τα χρώματα έτσι ώστε κάθε πόλη να έχει διαφορετικό χρώμα από τις γειτονικές της. **(β′)Χρωματισμός Γράφου:** Είναι η γενική κατηγορία του προβλήματος όπου χρησιμοποιείται η αφαιρετική αναπαράσταση των σχέσεων μεταξύ διαφορετικών κόμβων.



(α′) Map Coloring.



(β′) Graph Coloring.

Κεφάλαιο 2

Αλγόριθμοι Υπαναχώρησης

Μια πρώτη προσσέγγιση σ΄ αυτού του είδους τα προβλήματα είναι αυτή της αναζήτησης που στηρίζεται στην φύση των περιορισμών ενός προβλήματος και στην προεπεξεργασία τους με στόχο την αποτελεσματικότερη αναζήτηση. Τέτοιοι μηχανισμοί καλούνται **Αλγόριθμοι Υπαναχώρησης**. Παρακάτω αναφέρονται οι βασικότερες προσεγγίσεις αυτού του τύπου οι οποίες μπορούν να χρησιμοποιηθούν είτε μόνες τους είτε σε συνδιασμό με άλλες μεθόδους, αποτελώντας στάδιο προεπεξεργασίας του χώρου καταστάσεων.

2.1 Εξαντλητική Αναζήτηση τύπου Depth First Search

Η τεχνική DFS είναι η απλούστερη μέθοδος για την αντιμετώπιση τέτοιων προβλημάτων η οποία σπάνια επιλέγεται λόγω της πολύ κακής πολυπλοκότητας της. Αυτό που απλώς κάνει είναι ότι δημιουργεί όλους τους πιθανούς συνδιασμούς τιμών των μεταβλητών ενός προβλήματος δημιουργώντας έτσι όλες τις πιθανές καταστάσεις και για κάθε μία εξετάζεται το αν αποτελεί λύση. Αξίζει να σημειωθεί ότι αφού η πλήρης ανάθεση τιμών βρίσκεται στα φύλλα του

δέντρου αναζήτησης όπου γίνεται και ο έλεγχος, ο αλγόριθμος σπαταλά χρόνο αναζητώντας λύση ενώ κάποιοι περιορισμοί έχουν ήδη παραβιαστεί.

Η πολυπλοκότητα αυτής της μεθόδους είναι προφανώς ανάλογη με το πλήθος των μεταβλητών ενός προβλήματος καθώς και με το πλήθος των πεδίων τιμών κάθε μεταβλητής. Ο χρόνος χειρότερης περίπτωσης είναι αυτός στον οποίο η λύση βρίσκεται στον τελευταίο από το πλήρες σύνολο πιθανών καταστάσεων. Όσο λοιπόν αυξάνεται το μέγεθος ενός προβλήματος μεγαλώνει εκθετικά και ο χρόνος αναζήτησης λύσης.

2.2 Απλή Υπαναχώρηση (Backtracking Search)

Μία παραλλαγή του DFS αποτελεί η *Απλή ή Χρονολογική Υπαναχώρηση* (BT). Η λογική τής είναι πολύ απλή και γενική: Επιλέγουμε τιμές για μία μεταβλητή κάθε φορά, ελέγχουμε αν οι περιορισμοί ικανοποιούνται και υπαναχωρούμε όταν δεν υπάρχουν άλλες επιτρεπτές τιμές για να επιλεγούν. Η γενική ιδέα σ΄ αυτόν τον μηχανισμό είναι ότι δημιουργώντας μερικές λύσεις, προσπαθούμε να τις επεκτείνουμε σε μία πλήρη λύση.

Αυτή η τεχνική αν και πλήρης έχει πολύ υψηλή χρονική πολυπλοκότητα η οποία μπορεί να μειωθεί αν χρησιμοποιηθούν έξυπνες απαντήσεις σε ερωτήματα όπως: Ποιά θα είναι η επόμενη μεταβλητή που θα επιλεγεί, ποιές είναι οι συνέπειες της τρέχουσας ανάθεσης στις υπόλοιπες μεταβλητές που δεν έχουν τιμές και τέλος όταν μία διαδρομή αποτυγχάνει, μπορεί η αναζήτηση να αποφύγει την ίδια αποτυχία σε μελλοντικές διαδρομές;

2.3 Ευριστικοί Μηχανισμοί Επιλογής Μεταβλητών

Ο Ευριστικός Μηχανισμός Ελάχιστων Απομενουσών Τιμών (Minimum Remaining Values - MRV) απαντάει στο πρώτο ερώτημα προτείνοντας την επιλογή της μεταβλητής με τις λιγότερες επιτρεπτές τιμές από το πεδίο τιμών της με στόχο την ελαχιστοποίηση του παράγοντα διακλάδωσης στην πορεία της αναζήτησης. Αυτή η διαδικασία είναι δυναμική.

Ο Ευριστικός Μηχανισμός Βαθμού (Degree Heuristic) προτείνει την επιλογή της μεταβλητής που εμπλέκεται στον μεγαλύτερο πλήθος περιορισμών με άλλες μεταβλητές που δεν τους έχει ανατεθεί τιμή. Δηλαδή την μεταβλητή με το μεγαλύτερο βαθμό στον Γράφο Περιορισμών. Εδώ η διαδικασία είναι στατική.

Οι δύο παραπάνω τεχνικές συνήθως χρησιμοποιούνται συνδιαστικά. Όταν ο MRV δεν μπορεί να επιλέξει μεταβλητή λόγω ισοδυναμιών στο πλήθος των απομενουσών τιμών μεταξύ διαφορετικών μεταβλητών, χρησιμοποιείται ο Degree Heuristic.

2.4 Ευριστικοί Μηχανισμοί Επιλογής Τιμής

Από τη στιγμή που έχει επιλεγεί κάποια μεταβλητή, ο αλγόριθμος πρέπει ν΄ αποφασίσει με ποιά σειρά θα εξετάσει τις τιμές της. Ο ευριστικός μηχανισμός της Λιγότερο Δεσμευτικής Τιμής (Least Constraining Value) δίνει προτεραιότητα στην τιμή που αποκλείει τις λιγότερες επιλογές τιμών από τις γειτονικές μεταβλητές του Γράφου Περιορισμών. Παρ΄ όλα αυτά ο LCV δεν είναι χρήσιμος αν ενδιαφερόμαστε για όλες τις λύσεις ή αν το πρόβλημα που εξετάζεται δεν έχει λύση.

2.5 Πρώιμος Έλεγχος (Forward Checking)

Στην πορεία της αναζήτησης, ανατίθενται τιμές σε μεταβλητές με αποτέλεσμα να περιορίζεται έμμεσα το πεδίο τιμών των μεταβλητών που δεν τις έχει ανατεθεί τιμή. Αυτό συμβαίνει επειδή κάποιες τιμές μπορεί να είναι σε σύγκρουση με τις τιμές που έχουν ανατεθεί ήδη σε άλλες μεταβλητές. Το γεγονός αυτό, αν αξιοποιηθεί εγκαίρως σαν πληροφορία ή πριν ακόμη αρχίσει η αναζήτηση, μπορεί να περιορίσει δραστικά τον χώρο αναζήτησης.

Ο Πρώιμος Έλεγχος FC πετυχαίνει ακριβώς αυτό κάνοντας το εξής: Κάθε φορά που ανατίθεται τιμή σε μια μεταβλητή X, εξετάζεται κάθε μεταβλητή Y που συνδέεται με την X με έναν περιορισμό και διαγράφεται από το πεδίο τιμών της Y κάθε τιμή που είναι ασυνεπής με την τιμή που επιλέχθηκε για την X. Αξίζει να σημειωθεί ότι ο FC συνδιάζεται καλά με τον MRV.

2.6 Διάδοση Περιορισμών (Constraint Propagation)

Ο πρώιμος έλεγχος δεν καταφέρνει πάντοτε να ανιχνεύει όλες τις ασυνέπειες που μπορεί να προκύψουν. Για παράδειγμα, έστω ότι υπάρχουν 3 μεταβλητές x, y, z όπου η καθεμία έχει πεδίο τιμών το $\boldsymbol{D_x} = (1, 2, 3)$, $\boldsymbol{D_y} = (1, 2)$, $\boldsymbol{D_z} = (1, 2)$ και περιορισμούς τα: $x \neq y, x \neq z$ και $x \neq y$. Αν ανατεθεί η τιμή 1 στη x τότε ο FC θα διαγράψει από το πεδίο τιμών των y και z την τιμή 1 αφού μόνο αυτή συγκρούεται με την τιμή της x. Αφού τα νέα πεδία τιμών των y και z είναι τώρα $\boldsymbol{D} = (2)$, οι μοναδικές αναθέσεις που μπορούν να γίνουν σ΄ αυτές είναι $y = 2$ και $z = 2$. Αυτό όμως δεν μπορεί να συμβεί αφού πρέπει $y \neq z$. Παρ΄ όλα αυτά ο πρώιμος έλεγχος δεν μπόρεσε να το ανιχνεύσει.

Η Διάδοση Περιορισμών περιλαμβάνει τους μηχανισμούς που εξετάζουν και διαδίδουν ως πληροφορία τις επιπτώσεις ενός περιορισμού που ισχύει για μία με-

ταβλητή σε άλλες μεταβλητές. Έτσι, σε κάθε βήμα της αναζήτησης μπορούμε να «κλαδεύουμε» τμήματα του χώρου αναζήτησης, εξετάζοντας τις συνέπειες των μερικών αναθέσεων. Οι αλγόριθμοι που έχουν προταθεί γι΄ αυτό ονομάζονται αλγόριθμοι που βλέπουν μπροστά (look-ahead) και οι βασικότεροι από αυτούς αναφέρονται παρακάτω.

2.7 Συνέπεια Τόξου (Arc Consistency)

Η συνέπεια τόξου είναι μία γρήγορη μέθοδος για τη διάδοση των περιορισμών αρκετά ισχυρότερη από τον FC.

Ορισμός: Έστω X, Y μεταβλητές ενός CSP P και (X, Y) ένα κατευθυνόμενο τόξο στο γράφο περιορισμών του P. Η ακμή (X, Y) ονομάζεται συνεπής αν για κάθε τιμή x του X, υπάρχει μία τιμή y του Y έτσι ώστε η x να είναι συνεπής με την y.

Ορισμός: Ένα CSP ονομάζεται συνεπές ως προς τα τόξα (arc-consistent) αν όλα τα τόξα του γράφου περιορισμών του είναι συνεπή.

Κατά τη διάρκεια της αναζήτησης ή ως προεπεξεργασία αυτής, διαγράφονται οι τιμές που μπορούν να προκαλέσουν τέτοιου τύπου ασυνέπειες προσπαθόντας να καταστήσουν το CSP συνεπές ως προς τα τόξα κατά το μέγιστο δυνατό. Η πρώτη περίπτωση αναφέρεται ως αλγόριθμος MAC, από το *Maintaining Arc Consistency* - Διατήρηση Συνέπειας Τόξου και εκτελείται κάθε φορά που γίνεται μία ανάθεση.

2.8 k-συνέπεια

Τα CSP προβλήματα περιλαμβάνουν προβλήματα της κλάσης NP-Complete όπως για παράδειγμα το $3SAT$, οπότε ο έλεγχος για το αν ένα πρόβλημα ικανοποίησης περιορισμών είναι συνεπές αποτελεί κι αυτό ένα πρόβλημα εξίσου

δύσκολο. Έτσι συμπεραίνουμε ότι ο έλεγχος συνέπειας τόξου δεν εγγυάται την εύρεση και την εξάλειψη όλων των δυνατών ασυνεπειών.

Μια ισχυρότερη μορφή διάδοσης συνέπειας αποτελεί η k-συνέπεια.

Ορισμός: Ένα CSP είναι k-συνεπές εάν κάθε ανάθεση τιμών σε $k-1$ μεταβλητές $\{X_1 = v_1, \ldots, X_{k-1} = v_{k-1}\}$ που ικανοποιεί όλους τους περιορισμούς που εμπλέκουν τις μεταβλητές μπορεί να επεκταθεί σε μία ανάθεση $\{X_1 = v_1, \ldots, X_{k-1} = v_{k-1}, X_k = v_k\}$ που ικανοποιεί όλους τους περιορισμούς που εμπλέκουν τις μεταβλητές $X_1, \ldots, X_{k-1}, X_k$.

Για να γίνει ένα CSP k-συνεπές, εισάγουμε σ΄ αυτό νέους περιορισμούς με $k-1$ μεταβλητές που αποκλείουν τις αναθέσεις σε $k-1$ μεταβλητές που παραβιάζουν τον παραπάνω ορισμό.

Η 1-συνέπεια ή συνέπεια κόμβου όπως αλλιώς ονομάζεται, σ΄ ένα CSP σημαίνει ότι κάθε μεταβλητή X είναι συνεπής από μόνη της δηλαδή αν κάθε τιμή του πεδίου τιμών της δεν παραβιάζει έναν μοναδιαίο περιορισμό $c(X)$. Η 2-συνέπεια είναι η συνέπεια τόξου που αναφέρθηκε παραπάνω. 3-συνέπεια (συνέπεια διαδρομής) σημαίνει ότι οποιοδήποτε ζεύγος γειτονικών μεταβλητών μπορεί να είναι συνεπές με οποιαδήποτε τρίτη μεταβλητή.

Ένας γράφος περιορισμών έχει *ισχυρή k-συνέπεια* αν έχει k-συνέπεια και $(k-1)$-συνέπεια και $(k-2)$-συνέπεια ... και 1-συνέπεια. Ένα πρόβλημα που έχει ισχυρή k-συνέπεια μπορεί να επιλυθεί χωρίς καμία υπαναχώρηση αφού πάντα θα υπάρχουν τιμές για να επιλεγούν για μία μεταβλητή ώστε να υπάρχουν πάντα τιμές άλλων μεταβλητών που να είναι συνεπείς ώς προς αυτές.

2.9 Σύνοψη

Οι παραπάνω μέθοδοι που αναφέρονται αποτελούν μία πρώτη προσέγγιση των προβλημάτων ικανοποίησης περιορισμών. Μπορούν να χρησιμοποιηθούν ως προεπεξεργασία των προβλημάτων ώστε να διευκολύνουν την διαδικασία της

αναζήτησης. Η περαιτέρω ανάλυση αυτών των μεθόδων είναι θέμα που ξεφεύγει από τον στόχο αυτής της εργασίας. Η αναφορά τους παρόλα αυτά κρίθηκε σκόπιμη χάριν πληρότητας.

Κεφάλαιο 3

Τοπική Αναζήτηση

3.1 Εισαγωγή

Η Τοπική Αναζήτηση αποτελεί έναν διαφορετικό τρόπο προσέγγισης τέτοιων προβλημάτων. Η γενική φιλοσοφία είναι ότι κάθε αλγόριθμος τοπικής αναζήτησης ξεκινάει από μία πλήρη ανάθεση τιμών , συνήθως τυχαία και στη συνέχεια τη μεταβάλλει ή αλλιώς την επιδιορθώνει (heuristic repair) αλλάζοντας την τιμή μίας μεταβλητής κάθε φορά έως η κατάσταση να γίνει λύση εφόσον υπάρχει τέτοια (Σχήμα 3.1). Το σημαντικό πλεονέκτημα τέτοιων μεθόδων είναι η αποδοτικότητα ως προς τον χρόνο αναζήτησης λύσης αφού δεν εξετάζεται όλος ο χώρος καταστάσεων και ότι είναι εφαρμόσιμοι σε on-line περιβάλλοντα[1]. Μία αναζήτηση με υπαναχώρηση, με το νέο σύνολο περιορισμών που προκύπτει σε ένα on-line περιβάλλον, απαιτεί συνήθως πολύ περισσότερο χρόνο και θα μπορούσε να βρει μια λύση με πολλές αλλαγές σε σχέση με την αρχική κατάσταση. Οι αλγόριθμοι τοπικής αναζήτησης δεν είναι πλήρεις αφού δεν εγγυόνται την εύρεση λύσης αλλά η επιλογή τους για την αντιμετώπιση δύσκολων προβλημάτων πολλές φορές είναι μονόδρομος αφού στους πλήρεις όσο αυξάνεται το μέγεθος ή

[1]On-line περιβάλλον έχουν τα προβλήματα που αλλάζουν, δηλαδή σ΄ αυτά όπου μπορούν να μεταβάλλονται οι περιορισμοί ακόμη και κατά τη διάρκεια της αναζήτησης εύρεσης λύσης.

η δυσκολία ενός προβλήματος αυξάνεται εκθετικά και ο χρόνος εύρεσης λύσης κάτι το οποίο δεν ισχύει στους πρώτους.

Η Τοπική Αναζήτηση είναι αρκετά δημοφιλής επίσης σε προβλήματα βελτιστοποίησης όπως τα MAX-CSPs όπου το ζητούμενο είναι η ελαχιστοποίηση του κόστους της αντικειμενικής συνάρτησης στο μέγιστο δυνατό βαθμό. Το χαρακτηριστικό που την καθιστά χρήσιμη είναι ότι σε τέτοιου είδους προβλήματα η αναζήτηση διακόπτεται όταν συγκλίνει επαρκώς προς τη λύση κάτι το οποίο καθορίζεται εξαρχής.

Το πρόβλημα το οποίο συναντάται στους αλγόριθμους Τοπικής Αναζήτησης είναι ότι εύκολα παγιδεύονται κατά την διάρκεια της αναζήτησης σε τοπικές φαινομενικά καλές λύσεις. Αυτό συμβαίνει επειδή οι αλγόριθμοι αυτοί αποτελούν παραλλαγές αλγόριθμων αναρίχησης λόφων (Hill Climbing) όπου η αναζήτηση κινείται πάντοτε προς την κατεύθυνση όπου υπάρχει καλύτερη λύση, απορρίπτοντας τις κινήσεις που χειροτερεύουν το κόστος της αναζήτησης (Σχήμα 3.2).

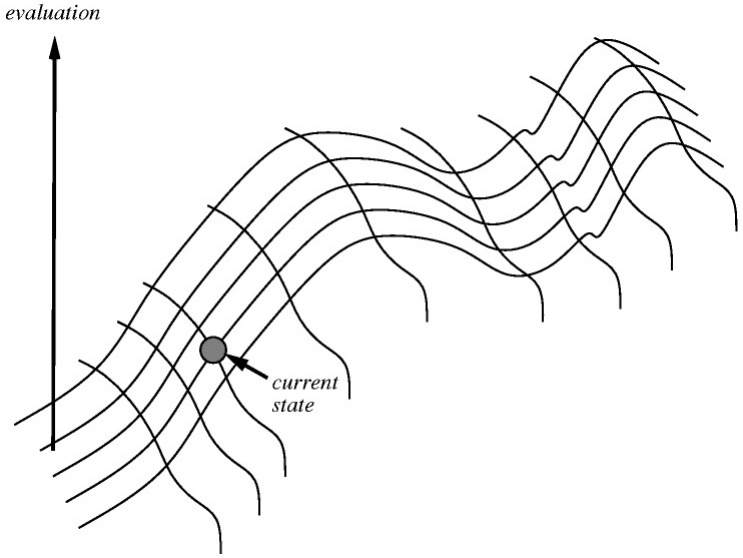
Ορισμοί:

Τοπικά Μέγιστα είναι οι καταστάσεις όπου είναι καλύτερες από κάθε γειτονική τους αλλά χειρότερες από το ολικό μέγιστο το οποίο αποτελεί συνήθως λύση.

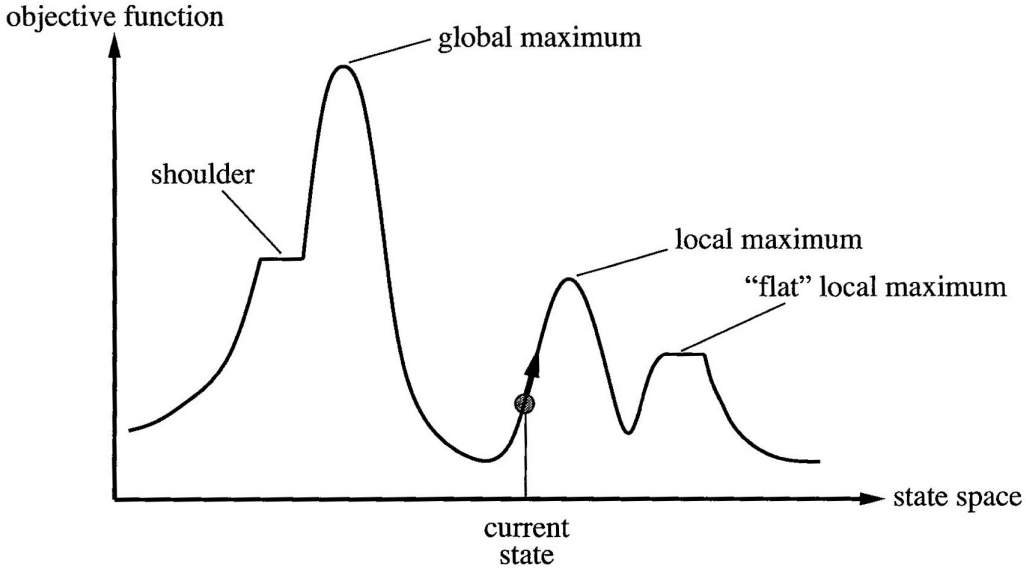
Κορυφογραμμές είναι η ύπαρξη ακολουθιακών τοπικών μεγίστων στα οποία όλες οι διαθέσιμες ενέργειες έχουν κατεύθυνση προς τα κάτω με αποτέλεσμα να δυσχεραίνουν την πλοήγηση της αναζήτησης.

Οροπέδια είναι η περιοχές όπου όλες οι γειτονικές καταστάσεις έχουν ίδιο κόστος. Μπορεί να μην υπάρχει διέξοδος προς τα πάνω ή να καταλήγουν σε ώμο από τον οποίο μπορεί να γίνει πρόοδος.

Σχήμα 3.1 Χώρος Καταστάσεων Προβλήματος



Σχήμα 3.2 Ακρότατα Αντικειμενικής Συνάρτησης



Πολλές διαφορετικές τεχνικές Τοπικής Αναζήτησης έχουν αναπτυχθεί ώστε να αντιμετωπίσουν τα προβλήματα της τοπικότητας. Στην παρούσα διπλωματική εργασία έγινε η υλοποίηση και ο πειραματικός έλεγχος της απόδοσης, σε διαφορετικές κλάσεις προβλημάτων, μερικών από τους βασικότερους των

αλγόριθμων Τοπικής Αναζήτησης καθώς και μια προσπάθεια υλοποίησης μίας δυναμικής μορφής της Ταμπού Αναζήτησης. Στις επόμενες ενότητες μελετώνται μία μία οι μέθοδοι που υλοποιήθηκαν.

Πρωτού αναλυθούν οι αλγοριθμικές τεχνικές που υλοποιήθηκαν κρίνεται σκόπιμη η αναφορά βασικών εννοιών οι οποίες χρησιμοποιούνται σε όλες τις μεθόδους.

Δομές Δεδομένων για την αναπαράσταση των προβλημάτων

Σε κάθε αλγόριθμο χρησιμοποιήθηκαν οι ίδιες δομές δεδομένων για την αναπαράσταση των προβλημάτων. Οι περιορισμοί ορίστηκαν ως ένας πίνακας δομή $Constraints[C]$, όπου C το πλήθος των περιορισμών που δίνεται κάθε φορά από το κάθε στιγμιότυπο προβλήματος. Κάθε στοιχείο του πίνακα $Constraints[]$ περιλαμβάνει δύο μεταβλητές x και y στις οποίες αφορά ο περιορισμός. Επίσης περιλαμβάνει έναν δυαδικό πίνακα $binConf[N][M]$, όπου N το πλήθος των στοιχείων του πεδίου τιμών της μεταβλητής x και M της y αντίστοιχα, ο οποίος λαμβάνει την τιμή 1 στις θέσεις όπου οι συντεταγμένες - τιμές των μεταβλητών αποτελούν περιορισμό και την τιμή 0 όπου τα ζεύγη των τιμών επιτρέπονται.

Ακόμη, ορίστηκε ένας πίνακας δομή $Dom[N]$, οπού N το πλήθος των διαφορετικών πεδίων τιμών του υπό εξέταση προβλήματος, κάθε στοιχείο του οποίου περιλαμβάνει έναν πίνακα $Domain[i]$ που περιλαμβάνει τις τιμές ενός πεδίου τιμών και μία μεταβλητή sz οι οποία λαμβάνει ως τιμή το πλήθος των στοιχείων του πεδίου τιμών.

Τέλος, ορίστηκε ένας πίνακας $V[N]$, όπου N το πλήθος των μεταβλητών, ο οποίος αναπαριστά τις μεταβλητές του εκάστοτε προβλήματος. Ο πίνακας V είναι συνεχώς γεμάτος με τιμές αφού στην τοπική αναζήτηση χρησιμοποιούνται μόνο πλήρεις καταστάσεις. Χάριν ευκολίας χρησιμοποιούμε τον συμβολισμό s όταν αναφερόμαστε στον πίνακα V για να εννοήσουμε μία πλήρη ανάθεση.

Συγκρούσεις

Όταν κάποιες μεταβλητές οι οποίες εμπλέκονται σε περιορισμούς έχουν τιμές οι οποίες ανήκουν ταυτόχρονα σε κάποιον περιορισμό τότε θεωρείται ότι βρίσκονται σε σύγκρουση (conflict). Για παράδειγμα έστω x και y δύο μεταβλητές οι οποίες έχουν πεδίο τιμών το σύνολο των θετικών ακεραίων. Υποθέτουμε ότι ορίζεται ως περιορισμός για αυτές το $y \neq 2x$. Αν το $x = 2$ και το $y = 4$ τότε λέμε ότι οι μεταβλητές x και y βρίσκονται σε σύγκρουση αφού παραβιάζουν τον περιορισμό.

Οι περιορισμοί που ορίζονται σε κάποιο πρόβλημα μπορούν να είναι μοναδιαίοι, δηλαδή να αφορούν μόνο σε μία μεταβλητή, δυαδικοί, όταν αφορούν σε δύο μεταβλητές, και γενικότερα $n-$αδικοί όταν αφορούν σε n μεταβλητές. Στην παρούσα εργασία έγινε μελέτη σε προβλήματα δυαδικών περιορισμών.

Συνάρτηση Αποτίμησης Κόστους (Evaluation Function)

Η συνάρτηση `eval(s)` η οποία χρησιμοποιείται σε όλους τους παρακάτω αλγόριθμους καλείται συνάρτηση αποτίμησης κόστους και μετράει το πλήθος των συγκρούσεων που υπάρχουν σε μία κατάσταση s. Όταν το πλήθος των συγκρούσεων σε μία κατάσταση s_0 είναι ίσο με μηδέν τότε θεωρούμε ότι η s_0 αποτελεί λύση του προβλήματος.

Για τον υπολογισμό των συγκρούσεων η `eval(s)` εξετάζει κάθε φορά έναν έναν τους περιορισμούς με βάση τις τιμές που έχουν οι μεταβλητές. Ποιο συγκεκριμένα για κάθε στοιχείο του πίνακα $Constraints[]$ εξετάζεται ο πίνακας $binConf[i][j]$ στη θέση που προσδιορίζουν οι συντεταγμένες οι οποίες αποτελούν τις τιμές των υπαρχουσών στον περιορισμό μεταβλητών. Για παράδειγμα αν ο περιορισμός $Constraints[1]$ περιλαμβάνει τις μεταβλητές x και y οι οποίες έχουν τιμές $x = 1$ και $y = 3$, τότε εξετάζεται η θέση του πίνακα $binConf[i][j]$ που έχει συντεταγμένες τα $i = 1$ και $j = 3$. Αν η τιμή αυτής της θέσης ισούται με 1 τότε οι μεταβλητές x και y βρίσκονται σε σύγκρουση και ο μετρητής των

συγκρούσεων αυξάνεται κατά 1.

Μέγιστος Αριθμός Βημάτων (Max Moves)

Οι ευριστικοί μηχανισμοί δεν εγγυόνται ότι θα βρούν λύση ούτε εξετάζουν ολόκληρο τον χώρο καταστάσεων ενός προβλήματος. Πολλές φορές ακόμη επισκέπτονται καταστάσεις τις οποίες έχουν ξαναεπισκεφθεί. Δεν εξασφαλίζεται λοιπόν η περατότητά τους. Για να μπορούν να τερματίζονται ορίζουμε ένα πλήθος μέγιστων βημάτων (ή αλλιώς επαναλήψεων) που μπορούν να εκτελεστούν. Αυτή η σταθερά ορίστηκε σε όλες τις περιπτώσεις $maxmoves = 500000$. Η μεταβλητή $nbmoves$ αποτελεί τον μετρητή επαναλήψεων του κάθε αλγόριθμου. Ξεκινάει από την τιμή μηδέν και αυξάνεται κατά 1 σε κάθε επανάληψη. Οι αλγόριθμοι συνεχίζουν να εκτελούνται όσο $nbmoves < maxmoves$.

3.2 Min-Conflicts

Ο ευριστικός μηχανισμός των *Ελάχιστων Συγκρούσεων* είναι μιά επισκευαστική μέθοδος που αξιοποιεί την τυχαιότητα στα πλαίσια της επιλογής μεταβλητής προς τροποποίηση με στόχο την ομαλή πλοήγηση προς την λύση.

Η γενική περιγραφή του είναι η εξής: Ξεκινάει κάνοντας μια τυχαία πλήρη ανάθεση s. Εξετάζεται το πλήθος των συγκρούσεων της τρέχουσας κατάστασης και εντοπίζονται οι μεταβλητές που εμπλέκονται σε συγκρούσεις. Έπειτα επιλέγεται από το σύνολο των σε σύγκρουση μεταβλητών μία μεταβλητή V στην οποία ανατίθεται η τιμή u' που ελαχιστοποιεί το πλήθος των συγκρούσεων παρακάμπτοντας τις ισοδυναμίες τυχαία. Αν δεν υπάρχει τέτοια τιμή τότε επιλέγεται οποιαδήποτε τιμή δεν αυξάνει το πλήθος των περιορισμών ενώ επιλέγεται η τρέχουσα τιμή u μόνο εφόσον όλες οι υπόλοιπες αυξάνουν το πλήθος των συγκρούσεων (Algorithm 1).

Αξίζει να σημειωθεί ότι η μέθοδος γίνεται εκπληκτικά αποδοτική όταν της

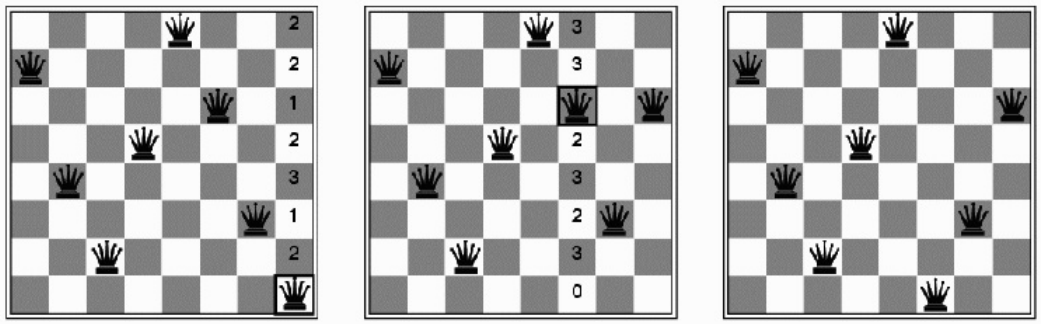
δοθεί μια λογική αρχική κατάσταση αντί της τυχαίας κάτι το οποίο αποφεύχθηκε για λόγους αμερόληπτης πειραματικής σύγκρισης με τους υπόλοιπους αλγόριθμους.

Δυστυχώς, ο MC όπως και κάθε άλλος αλγόριθμος αναρρίχησης λόφων, όταν οι λύσεις του προβλήματος δεν είναι πυκνά κατανεμημένες στον χώρο καταστάσεων, παγιδεύεται εύκολα σε τοπικά βέλτιστα με αποτέλεσμα να μην μπορεί να κάνει βελτιωτική κίνηση.

Η απόδοση του MC είχε ελεγχθεί για πρώτη φορά στο πρόβλημα των *N-Βασιλισσών* από τον Mark D. Johnston ο οποίος τον υλοποίησε με στόχο την χρονοδρομολόγηση των παρατηρήσεων του τηλεσκόπιου Hubble [3]. Στόχος σε αυτό το πρόβλημα είναι η τοποθέτηση 8 βασιλισσών πάνω στη σκακιέρα έτσι ώστε να μήν απειλείται καμία βασίλισσα. Ο MC επιλύει το πρόβλημα επιλέγοντας τυχαία μία στήλη στην οποία βρίσκεται μια βασίλισσα που βρίσκεται υπο απειλή, και τοποθετώντας τη στο τετράγωνο με τις λιγότερες συγκρούσεις, δηλαδή στο σημείο όπου η βασίλισσα απειλείται ή απειλεί τις λιγότερες βασίλισσες (Σχήμα 3.4). Το πρόβλημα γενικεύεται σε ταμπλό $N \times N$ όπου και γίνεται δυσκολότερο. Παρ΄ όλα αυτά, ο χρόνος εκτέλεσης του αλγόριθμου είναι ανεξάρτητος του μεγέθους του προβλήματος. Ο MC επιλύει το πρόβλημα του 1 εκατομυρίου βασιλισσών σε κατά μέσο όρο 50 κινήσεις. Από αυτά τα αποτελέσματα το 1990 άνοιξε ο δρόμος προς την έρευνα σε προβλήματα τοπικής αναζήτησης και του διαχωρισμού εύκολων και δύσκολων προβλημάτων.

Ο αλγόριθμος πέρα από την ικανότητά του να επιλύει εύκολα προβλήματα για την τοπική αναζήτηση όπως το N-Queen Problem αποδείχθηκε ικανός και σε δύσκολα προβλήματα. Για παράδειγμα στο Επιστημονικό Ίδρυμα Διαστημικού Τηλεσκοπίου (Space Telescope Science Institute) του Πανεπιστημίου Johns

Σχήμα 3.4 Διαδοχικές κινήσεις του MC για την επίλυση του προβλήματος των N-Βασιλισσών



Hopkins University της Βαλτιμόρης έγινε μελέτη πάνω σ΄ ένα νευρωνικό δίκτυο όπου με τη χρήση του MC κατάφεραν να μειώσουν τον χρόνο παρατήρησης του τηλεσκόπιου Hubble από 3 εβδομάδες σε μόλις περίπου 10 λεπτά.

Algorithm 1 `Min-Conflicts (P: a CSP, maxmoves:the number of moves allowed, V: a variable from P, u:the current value of V)`

1: $s \leftarrow$ random complete variable assignment
2: $nbmoves \leftarrow 0$
3: **while** `eval`$(s) > 0$ AND $nbmoves < maxmoves$ **do**
4: Choose randomly a conflicting variable V
5: Choose a value u' that minimizes the constraint violations for V
6: **if** $u' \neq u$ **then**
7: assign u' to V
8: $nbmoves \leftarrow nbmoves + 1$
9: **end if**
10: **end while**
11: **return** s

3.3 Min-Conflicts - Random Walk

Οι αλγόριθμοι που βασίζονται στον MC, στηρίζονται απλώς στις επαναλήψεις του ευριστικού μηχανισμού MC. Η ικανότητα επίλυσης τέτοιων αλγόριθμων είναι περιορισμένη αφού ο MC δεν μπορεί να υπερβεί τα τοπικά βέλτιστα. Η απόδοση ενός τέτοιου αλγόριθμου όμως μπορεί να αυξηθεί αρκετά αν εισαχθούν κάποιες τεχνικές «θορύβου».

Μια από τις πιο δημοφιλείς στρατηγικές θορύβου είναι αυτή της Τυχαίας Περιήγησης (Random Walk). Ο συνδιασμός της με τον MC παράγουν μία υβριδική παραλλαγή MCRW η οποία περιγράφεται απλά ως εξής: Για μια δεδομένη μεταβλητή εμπλεκόμενη σε σύγκρουση, ανάθεσε μία τυχαία τιμή από το πεδίο τιμών της με πιθανότητα p αλλιώς με πιθανότητα $1-p$ εφάρμοσε τον MC ευριστικό μηχανισμό (Algorithm 2).

Algorithm 2 `Min-Conflicts + Random Walk (P: a CSP, maxmoves:the number of moves allowed, V: a variable from P, u:the current value of V, p:the probability for applying RW strategy)`

1: $s \leftarrow$ random complete variable assignment
2: $nbmoves \leftarrow 0$
3: **while** `eval`$(s) > 0$ AND $nbmoves < maxmoves$ **do**
4: **if** probability p verified **then**
5: Choose randomly a conflicting variable V
6: Choose randomly a value u' for V
7: **else**
8: Choose randomly a conflicting variable V
9: Choose a value u' that minimizes the constraint violations for V
10: **end if**
11: **if** $u' \neq u$ **then**
12: assign u' to V
13: $nbmoves \leftarrow nbmoves + 1$
14: **end if**
15: **end while**
16: **return** s

Αξίζει να σημειωθεί ότι η παράμετρος p διαδραματίζει καθοριστικό ρόλο στην απόδοση του αλγόριθμου. Προκαταρκτικές μελέτες έδειξαν ότι το εύρος των επιτρεπτών τιμών για το p είναι το $0.02 \leq p \leq 0.1$. Περαιτέρω αναφορά στη ρύθμιση των παραμέτρων της παρούσας υλοποίησης γίνεται στο Κεφάλαιο 4.

Η τυχαία περιήγηση επιτρέπει στον αλγόριθμο να κάνει μη βελτιωτικές κινήσεις ή ακόμη και κινήσεις που αυξάνουν το πλήθος των συγκρούσεων. Έτσι, ο αλγόριθμος μπορεί να ξεφύγει από κάποιο τοπικό βέλτιστο όταν η αναζήτηση συγκλίνει προς αυτό.

3.4 Tabu Search

Η Αναζήτηση Ταμπού είναι ένας μετα-ευριστικός[2] μηχανισμός επαναληπτικής τοπικής αναζήτησης. Το ιδιαίτερο χαρακτηριστικό της σε σύγκριση με τους υπόλοιπους μετα-ευριστικούς μηχανισμούς είναι η συστηματική χρήση της μνήμης για την καθοδήγηση της διαδικασίας της αναζήτησης.

Η πιο κοινή εκδοχή της TS είναι αυτή που κάνει χρήση μιας βραχυπρόθεσμης μνήμης (short-term memory) ώστε να δραπετεύει από τα τοπικά βέλτιστα. Τυπικά, η TS είναι μία «επιθετική» τοπική αναζήτηση η οποία σε κάθε βήμα προσπαθεί να κάνει την καλύτερη δυνατή κίνηση από την τρέχουσα κατάσταση s σε μία γειτονική κατάσταση s' ακόμη και αν αυτή χειροτερεύει την τιμή της αντικειμενικής συνάρτησης $f(s)$. Για να αποτρέψει την επιστροφή της τοπικής αναζήτησης στις αμέσως προηγούμενες λύσεις και γενικότερα να αποφευχθεί η επαναφορά στις ίδιες περιοχές λύσεων, η TS απαγορεύει κινήσεις που οδηγούν στις πρόσφατα επισκεφθείσες λύσεις. Αυτό επιτυγχάνεται με την αποθήκευση των πρόσφατα επισκεφθεισών λύσεων στη μνήμη και απογορεύοντας ρητώς την κίνηση προς αυτές.

Η συνηθέστερη τεχνική που χρησιμοποιείται είναι αυτή κατά την οποία τα ζεύγη $< V, u >$ που αποτελούν κινήσεις ταμπού, απογορεύεται να εισαχθούν στην τρέχουσα κατάσταση s. Πιο συγκεκριμένα, οι πρόσφατες λύσεις απαγορεύονται για tl επαναλήψεις. Η παράμετρος tl ονομάζεται tabu tenure και προσδιορίζει ουσιαστικά το μέγεθος της λίστας Ταμπού (Tabu List). Λίστα Ταμπού καλείται η λίστα η οποία περιλαμβάνει ζεύγη μεταβλητής - τιμής $< V, u >$ τα οποία απαγορεύεται να εισαχθούν στην s για tl επαναλήψεις. Η απαγόρευση

[2]Μετα-ευριστικός μηχανισμός είναι μία γενική μέθοδος που περιλαμβάνει ένα σύνολο κανόνων και στρατηγικών για την επίλυση προβλημάτων γενικής φύσεως όπως για παράδειγμα τα προβλήματα βελτιστοποίησης. Ένας μετα-ευριστικός μηχανισμός μπορεί να προσαρμοστεί και να εξειδικευτεί στα πλαίσια ενό ευριστικού μηχανισμού που επιλύει ένα συγκεκριμένο πρόβλημα.

συγκεκριμένων ζευγών ισοδυναμεί με τον δυναμικό περιορισμό της γειτονιάς $N(s)$ της τρέχουσας λύσης s σε ένα υποσύνολο της U αποδεκτών καταστάσεων. Γι΄ αυτό το λόγο η Αναζήτηση Ταμπού μπορεί να θεωρηθεί τεχνική αναζήτησης δυναμικής γειτονιάς. Η έκδοση της TS που υλοποιήθηκε στην παρούσα διπλωματική εργασία είναι αυτή των Galinier και Hao η οποία είχε θεωρηθεί και η πιο αποδοτική [4]. Ο ψευδοκώδικας του αλγόριθμου παρουσιάζεται παρακάτω Algorithm 3.

Οι συνθήκες ταμπού σε ορισμένες περιπτώσεις μπορεί να καταστούν ιδιαίτερα περιοριστικές για την αναζήτηση, εμποδίζοντας την αναζήτηση από το να επισκεφτεί «ελκυστικές» λύσεις ή καταστάσεις τις οποίες δεν έχει ξαναεπισκεφτεί. Αυτό το πρόβλημα αντιμετωπίζεται με την εισαγωγή ειδικών κριτηρίων αφαίρεσης της ταμπού ιδιότητας από συγκεκριμένες κινήσεις όταν αυτό είναι αναγκαίο. Τα κριτήρια αυτά καλούνται *Κριτήρια Φιλοδοξίας* (*Aspiration Criteria*) και το συνηθέστερο εξ αυτών ορίζεται με τον εξής τρόπο: Όταν η επιλογή ενός ζεύγους $< V, u >$,το οποίο αποτελεί ταμπού, βελτιώνει την τιμή της αντικειμενικής συνάρτησης $f(s)$ περισσότερο από την έως τότε βέλτιστη που έχει βρεθεί και είναι η καλύτερη από το σύνολο όλων των πιθανών κινήσεων, επιτρεπτών και μη, τότε αυτή επιλέγεται ως επόμενη κίνηση παρακάμπτοτας την ιδιότητα της ως ταμπού.

Algorithm 3 `TS-GH (P: a CSP, tl: the tabu tenure, maxmoves: the number of moves allowed)`

1: $s \leftarrow$ random complete variable assignment
2: $nbmoves \leftarrow 0$
3: initialize randomly the tabu list
4: **while** `eval`$(s) > 0$ AND $nbmoves < maxmoves$ **do**
5: choose the variable-value pair $< V, u' >$ that minimizes constraint violations, where V is a conflicting variable
6: **if** $< V, u' >$ is not tabu **or** it satisfies the aspiration criterion **then**
7: introduce $< V, u >$ in the tabu list, where u is the current value of V
8: remove the oldest variable-value pair from the tabu list
9: assign u' to V
10: $s \leftarrow$ new complete variable assignment
11: $nbmoves \leftarrow nbmoves + 1$
12: **end if**
13: **end while**
14: **return** s

Η TS οδηγείται προς την λύση με επιθετικό τρόπο. Έτσι, απαιτούνται ειδικές δομές δεδομένων ώστε να επιταχύνεται η διαδικασία εκτίμησης των κινήσεων (αν είναι ταμπού ή όχι) καθώς και να μειώνονται οι απαιτήσεις για την εύρεση της βέλτιστης κίνησης.

Ορισμός της συνάρτησης γειτνίασης

Για τον καθορισμό των γειτονικών καταστάσεων της τρέχουσας κατάστασης s χρησιμοποιήθηκε μία συνάρτηση γειτνίασης $N(s)$ στην οποία ορίζεται ότι: Για κάθε κατάσταση s από τον πλήρη χώρο καταστάσεων S, η $s' \in N(s)$ αν και μόνο αν η s' διαφέρει από την s κατά την τιμή μόνο μίας μεταβλητή που είναι

εμπλεκόμενη σε σύγκρουση.

Με άλλα λόγια, η αλλαγή τιμής σε μία μόνο μεταβλητή που εμπλέκεται σε σύγκρουση, παράγει μία γειτονική κατάσταση s' της s. Αξίζει να σημειωθεί ότι το μέγεθος του συνόλου των γειτονικών καταστάσεων μεταβάλλεται κατά την πορεία της αναζήτησης αφού το πλήθος των συγκρούσεων είναι μεταβαλλόμενο.

Λίστα Ταμπού και προσδιορισμός των λοιπών χαρακτηριστικών της Αναζήτησης Ταμπού

Κίνηση αποτελεί η αλλαγή τιμής u μίας μεταβλητής V που είναι σε διένεξη, με μία τιμή u' και γι΄ αυτό χαρακτηρίζεται από ένα ζεύγος $< V, u >$. Συνεπώς, όταν η λύση s αλλάζει σε s' με την αντικατάσταση της τρέχουσας τιμής u της V με μία νέα u', το ζεύγος $< V, u >$ χαρακτηρίζεται ταμπού για της επόμενες tl επαναλήψεις. Αυτό συνεπάγεται ότι η u δεν μπορεί να ανατεθεί ξανά στην V μέσα σ΄ αυτή τη περίοδο. Όπως η πιθανότητα p για τον MCRW, έτσι και η παράμετρος tl έχει μεγάλη επίδραση στην απόδοση της Ταμπού Αναζήτησης και μελέτες έδειξαν ότι η τιμή της μπορεί να ανήκει στο $10 < tl < 30$.

Για την υλοποίηση της λίστας ταμπού, χρησιμοποιήθηκε ένας $|X| \times |D|$ πίνακας T. Όταν ένα ζεύγος $< V, u >$ επιλέγεται ώς κίνηση, η θέση του πίνακα T που αντιστοιχεί σε αυτό λαμβάνει ως τιμή του άθροισμα του αριθμού της τρέχουσας επανάληψης συν το tl. Με αυτόν τον τρόπο είναι εύκολο να διαπιστώσουμε αν μία κίνηση είναι ταμπού απλά συγκρίνοντας τον τρέχοντα αριθμό επανάληψης με αυτόν που είναι αποθηκευμένος στον T.

Επιπροσθέτως, υπάρχουν και άλλες ενδιαφέρουσες τεχνικές που μπορούν να βελτιώσουν την απόδοση του αλγόριθμου όπως οι μέθοδοι *εντατικοποίησης (intensification)* και *διαφοροποίησης (diversification)* από τις οποίες η δεύτερη χρησιμοποιήθηκε σε μία παραλλαγή της (TS) και παρουσιάζεται στην Ενότητα 3.7.

3.5 Local Beam Search

Οι αλγόριθμοι αναρρίχησης λόφων χρησιμοποιούν μία εξαιρετικά επιθετική αναζήτηση με αποτέλεσμα να παγιδεύονται εύκολα σε τοπικά μέγιστα ιδίως όταν αντιμετωπίζουν ένα NP-hard πρόβλημα το οποίο συνήθως έχει εκθετικά υψηλό αριθμό τοπικών βέλτιστων. Η αναρρίχηση λόφων με τυχαίες επανεκκινήσεις από την άλλη πλευρά είναι θεωρητικά πλήρης αν υποθέσουμε ότι μπορεί να κάνει άπειρες επανεκκινήσεις. Σε δύσκολα προβλήματα όμως, τα οποία έχουν αρκετά λίγες λύσεις συγκριτικά με τον συνολικό χώρο καταστάσεων, οι επανεκκινήσεις μπορεί να απομακρύνουν την αναζήτηση συνεχώς από τις λύσεις και παρά το γεγονός ότι μια κατάσταση στόχου μπορεί να τύχει ως αρχική, την καθιστούν ιδιαίτερα χρονοβόρα. Ακόμη από κάθε αποτυχημένη προσπάθεια εύρεσης λύσης δεν εξάγεται κάποια πληροφορία που να καθοδηγεί την αναζήτηση στην πορεία των επανεκκινήσεων του αλγόριθμου.

Η Τοπική Ακτινική Αναζήτηση αποτελεί έναν ευριστικό μηχανισμό που συνδιάζει την αναρρίχηση λόφων με την Ακτινική Αναζήτηση (Beam Search) κάνοντας χρήση μνήμης σταθερού μεγέθους. Η περιγραφή του είναι η εξής: Ξεκινά με k τυχαία παραγόμενες καταστάσεις και παράγει όλους τους διαδόχους αυτών. Αν κάποια από τις k ή των διαδόχων τους αποτελεί λύση σταματάει και

την επιστρέφει. Αλλιώς, επιλέγει τους k καλύτερους διαδόχους από ολόκληρη την λίστα και επαναλαμβάνεται μέχρι να βρει λύση ή να ξεπεράσει κάποιον προκαθορισμένο αριθμό επαναλήψεων.

Το σημαντικό πλεονέκτημα του αλγόριθμου είναι ότι μεταβιβάζονται χρήσιμες πληροφορίες μεταξύ των k παράλληλων νημάτων της αναζήτησης. Έτσι, η αναζήτηση προσανατολίζεται προς ελκυστικές περιοχές καταστάσεων εγκαταλείποντας γρήγορα τις «άγονες». Για παράδειγμα αν μία από τις k αρχικές καταστάσεις είναι αρκετά καλύτερη από τις υπόλοιπες τότε πιθανότατα έχει και πολύ καλύτερους διαδόχους η οποίοι θα επιλεγούν ως η νέα γενιά k καταστάσεων μεταφέροντας τους πόρους εκεί όπου υπάρχει μεγαλύτερη πρόοδος.

Ένα εύλογο ερώτημα που προκύπτει είναι το ποιες τιμές είναι κατάλληλες για το k και αν αυτό είναι ανάλογο του μεγέθους του εκάστοτε προβλήματος. Για $k = 1$ ο αλγόριθμος μετατρέπεται σε απλή αναρρίχηση λόφων ενώ για πολύ μεγάλο k έχουμε το φαινόμενο της *Υπερπροσαρμογής* (Overfitting) οπού η απόδοση του αλγόριθμου πλέον καθορίζεται από το μεγάλο πλήθος των περιοχών που εξετάζονται και όχι από τον αλγόριθμο καθ' αυτό. Στη δεύτερη περίπτωση ακόμη, η χρονική και η χωρική πολυπλοκότητα του αλγόριθμου αυξάνονται εκθετικά μειώνοντας δραματικά την απόδοση του. Τελικά η τιμή που χρησιμοποιήθηκε προέκυψε πειραματικά βάσει των διαφορετικών κλάσεων προβλημάτων που εξετάστηκαν και ορίστηκε στο εύρος $10 < k < 20$. Επιπλέον πληροφορίες για ρύθμιση της παραμέτρου βρίσκονται στο Κεφάλαιο 4 - Πειραματική Διαδικασία. Ο ψευδοκώδικας του αλγόριθμου παρουσιάζεται παρακάτω (Algorithm 4).

Algorithm 4 `LBS (P: a CSP, k: number of initial states, maxmoves: the number of moves allowed, s:the state with the lowest` $f(s)$ `from` k`)`

1: $nbmoves \leftarrow 0$
2: initialize randomly k starting states
3: **while** `eval`$(s) > 0$ AND $nbmoves < maxmoves$ **do**
4: `Successors(`k`)`, generate all successors from all k states
5: Select the k best successors
6: **end while**
7: **return** s

Δομές Δεδομένων και λεπτομέρειες υλοποίησης

Για την υλοποίηση της λίστας διαδόχων χρησιμοποιήθηκε δομή δεδομένων τύπου Ουράς Προτεραιότητας (Priority Queue) με κλειδί κάθε κόμβου s το κόστος $f(s)$. Ορίστηκαν δύο συνδεδεμένες λίστες, μία κλειστή και μία ανοιχτή. Στην ανοιχτή παράγονται από την συνάρτηση `Successors()` και τοποθετούνται οι διάδοχοι των k καταστάσεων ενώ στην κλειστή τοποθετούνται οι k διάδοχοι που επιλέγονται για την επόμενη γενιά με ταξινόμηση κατά την εισαγωγή. Σε κάθε επανάληψη οι κόμβοι της ανοιχτής λίστας διαγράφονται ώστε η λίστα να υποδεχθεί τους καινούριους διαδόχους. Η παραπάνω μέθοδος, αν και αναγκαία, αυξάνει αρκετά την πολυπλοκότητα του αλγόριθμου συγκριτικά με τους υπόλοιπους ευρετικούς μηχανισμούς που υλοποιήθηκαν και αυτό φαίνεται και στα αποτελέσματα της πειραματικής διαδικασίας στο Κεφάλαιο 4.

3.6 Stochastic Local Beam Search

Μερικές φορές ανάλογα με την δομή του προβλήματος που τίθεται προς επίλυση, η LBS μπορεί να περιορίσει την αναζήτηση σε μία μικρή περιοχή του

χώρου καταστάσεων όπου θα μπορούσε ούτως ή άλλως να καταλήξει κάποιος απλός αλγόριθμος αναρρίχησης λόφων. Αυτό το πρόβλημα το περιορίζει μια παραλλαγή του αλγόριθμου η οποία καλείται Στοχαστική Ακτινική Αναζήτηση (Stochastic Local Beam Search). Η SLBS αντί να διαλέγει τους k καλύτερους από την δεξαμενή των υποψήφιων διαδόχων, διαλέγει k διαδόχους στην τύχη με τον καθένα να έχει πιθανότητα να επιλεγεί ανάλογη της αξίας του.

Πιο συγκεκριμένα σε κάθε επανάληψη υπολογίζεται το πλήθος $sat(s)$ των περιορισμών που ικανοποιούνται σε κάθε υποψήφιο διάδοχο που βρίσκεται στη λίστα. Αυτό αποτελεί το μέτρο συνέπειας (satisfiability) της κάθε κατάστασης s. Στη συνέχεια εντοπίζεται το sat_{min} και το sat_{max} και με βάση αυτά γίνεται κανονικοποίηση τύπου $min-max$ των δεδομένων στο εύρος $0 < p < 1$. Τελικά το p προκύπτει από τον τύπο $p = \frac{sat(s)-sat_{min}}{sat_{max}-sat_{min}}$. Κάθε φορά που εξετάζεται ένας διάδοχος, έχει πιθανότητα p να επιλεχθεί και αυτό συνεχίζεται μέχρι να επιλεγούν συνολικά k καταστάσεις. Κατά αυτόν τον τρόπο η αναζήτηση περιορίζεται σε μία μικρή περιοχή λύσεων μόνο όταν το $r \to 0$, όπου r το εύρος $sat_{max} - sat_{min}$. Αυτό συμβαίνει γιατί όταν το sat_{max} διαφέρει πολύ λίγο από το sat_{min} η κανονικοποίηση καθορίζει την πιθανότητα p των καταστάσεων σε ακραίες τιμές στο διάστημα $[0, 1]$. Έτσι, οι πρώτες καταστάσεις θα έχουν πιθανότητα p κοντά στο 1 και επειδή το $k << N(s)$ θα επιλέγονται πάντα μόνο αυτές.

Algorithm 5 `SLBS (P: a CSP, k: number of initial states, maxmoves: the number of moves allowed, s:the state with the lowest` $f(s)$ `from` k`)`

1: $nbmoves \leftarrow 0$
2: initialize randomly k starting states
3: **while** `eval`$(s) > 0$ **and** $nbmoves < maxmoves$ **do**
4: `Successors(`k`)`, generate all successors from all k states
5: `Normalize(S)`, normalize states according to their satisfiability
6: $n \leftarrow 0$
7: **while** $n < k$ **do**
8: **if** possibility $P(s_i)$ is verified **then**
9: select s_i as a successor
10: $n \leftarrow n + 1$
11: **end if**
12: **end while**
13: **end while**
14: **return** s

3.7 Dynamic Tabu Search

Η Αναζήτηση Ταμπού αποτελεί έναν από τους πιο αποτελεσματικούς μεταευριστικούς μηχανισμούς για την επίλυση προβλημάτων ικανοποίησης περιορισμών κάτι το οποίο φαίνεται και στ΄ αποτελέσματα της πειραματικής διαδικασίας. Η ικανότητά της να δραπετεύει από τα τοπικά βέλτιστα έγκειται στο γεγονός ότι διατηρεί το πρόσφατο ιστορικό αναζήτησης στη μνήμη ώστε να μην ξαναεπισκέπτεται άμεσα αυτές τις περιοχές. Η ρύθμιση της παραμέτρου tl καθορίζει πόσα ζεύγη μεταβλητής - τιμής $< V, u >$ αποτελούν κάθε χρονική στιγμή ταμπού. Αυτό όμως είναι κάτι που καθορίζεται πρωτού ξεκινήσει η αναζήτηση

με αποτέλεσμα το μέγεθος της λίστας ταμπού να παραμένει σταθερό κατά τη διάρκειά της. Η τιμή της tl παίζει καθοριστικό ρόλο στην αποδοτικότητα του αλγόριθμου. Μία μικρή τιμή έχει ως αποτελέσμα η αναζήτηση να περιφέρεται ατέρμονα γύρω από μία μικρή περιοχή λύσεων ενώ μία μεγάλη τιμή μπορεί να λειτουργήσει αρκετά περιοριστικά.

Κατά τη διάρκεια της αναζήτησης και σε ορισμένα προβλήματα αυτό μπορεί να συμβεί ακόμη και αν η tl οριστεί σε μία καλή αρχική τιμή. Για παράδειγμα αυτό συμβαίνει συνήθως όταν η αναζήτηση συναντάει ένα τοπικό βέλτιστο γύρω από το οποίο εκτείνεται μιά μεγάλη περιοχή με χειρότερες καταστάσεις. Έτσι, ενώ η λίστα ταμπού ανανεώνεται πλήρως με καινούρια ζεύγη όταν η αναζήτηση κάνει επαναλήψεις περισσότερες σε πλήθος από το μέγεθος της λίστας και δεν έχει πλησιάσει περιοχή όπου βρίσκεται κάποιο άλλο τοπικό ή ολικό βέλτιστο, οι κινήσεις που βελτιώνουν το κόστος είναι αυτές που επαναφέρουν την αναζήτηση στην κατεύθυνση του αρχικού τοπικού βέλτιστου. Μ' αυτό τον τρόπο η αναζήτηση μπορεί να παρομοιαστεί με τη συμπεριφορά ενός καραβιού που πάει να αποφύγει μία δίνη αλλά αυτή το ξανατραβάει πίσω.

Δική μας πρόταση για την αντιμετώπιση των ανωτέρω προβλημάτων αποτελεί μία παραλλαγή της Αναζήτησης Ταμπού την οποία καλούμε Δυναμική Αναζήτηση Ταμπού (Dynamic Tabu Search). Η βασική ιδέα είναι ότι κατά τη διάρκεια της αναζήτησης το μέγεθος της λίστας ταμπού μεταβάλλεται ανάλογα με τις εκάστοτε συνθήκες. Όταν η αναζήτηση δεν γυρίζει γύρω από τις ίδιες περιοχές για κάποιο αριθμό επαναλήψεων τότε η λίστα ταμπού μπορεί σταδιακά να μικραίνει. Όποτε η αναζήτηση περιφέρεται σε μία περιοχή λύσεων για μεγάλο αριθμό βημάτων τότε η λίστα πρέπει να μεγαλώσει.

Στην παρούσα υλοποίηση η παράμετρος tl αρχικά λαμβάνει μία τυχαία τιμή στο εύρος $10 < tl < 30$. Έπειτα, ξεκινάει ο ευριστικός μηχανισμός της Αναζήτησης Ταμπού και κάθε φορά που εντοπίζεται καλύτερη λύση, αυτή αποθηκεύεται στη μνήμη καθώς και ο αριθμός των συγκρούσεων $f(s)$ που υπάρχουν

σ΄ εκείνη. Το $f(s)$ κάθε χρονική στιγμή προσδιορίζει επίσης και των μικρότερο αριθμό συγκρούσεων f_{min} που έχουν εντοπιστεί μέχρι εκείνη τη στιγμή. Ποιο συγκεκριμένα, η κατάσταση αυτή αποθηκεύεται σε μία άλλη λίστα F και παραμένει εκεί έως ότου βρεθεί κάποια καλύτερη. Αν εντοπιστεί κατάσταση με ίδιο αριθμό συγκρούσεων f_{min}, αποθηκεύεται κι αυτή στην ίδια λίστα. Μόλις βρεθεί καλύτερη λύση, δηλαδή με μικρότερο $f(s)$, η λίστα αδειάζει και τοποθετείται σ΄ αυτή η νέα κατάσταση και η καινούρια τιμή f_{min}. Με άλλα λόγια στη λίστα F αποθηκεύονται οι βέλτιστες καταστάσεις που έχουν εντοπιστεί και έχουν όλες την ίδια τιμή f_{min}. Η λίστα αυτή έχει μέγιστο μέγεθος ίσο με το πενταπλάσιο μέγεθος της λίστας ταμπού tl ώστε να μην αυξάνεται πολύ ο χρόνος ελέγχου και να μην εξαντλείται η μνήμη. Αυτό καθορίστηκε πειραματικά αφού φάνηκε να είναι ένα επαρκές μέγεθος τουλάχιστον για τις κλάσεις προβλημάτων που χρησιμοποιήθηκαν στην πειραματική διαδικασία.

Κάθε φορά που η αναζήτηση επισκέφτεται μία λύση που έχει κόστος ίσο με το ελάχιστο f_{min}, εξετάζεται το αν υπάρχει μέσα στη λίστα των βέλτιστων καταστάσεων. Αν υπάρχει, σημαίνει ότι η αναζήτηση επανήλθε σε μία περιοχή λύσεων που είχε επισκεφτεί προηγουμένως και η λίστα ταμπού αυξάνεται κατά $tl' = tl + \frac{|X|\times|D|}{100}$ ώστε αυτό να αποτραπεί μελλοντικά[3]. Αν η νέα βέλτιστη κατάσταση δεν βρίσκεται στη λίστα τότε προστίθεται σ΄ αυτή και το μέγεθος της λίστας ταμπού μειώνεται σταδιακά ώστε να μην περιορίζεται η αναζήτηση. Η μείωση αυτή ορίζεται ως $tl' = tl - 1$ και εφαρμόζεται εφόσον το μέγεθος της λίστας ταμπού είναι μεγαλύτερο του 1. Επίσης για να αποτραπεί η απότομη μείωση ή αύξηση του μεγέθους της λίστας τέθηκε ως όρος να ξαναεπισκεφτεί η αναζήτηση $5tl$ τοπικά βέλτιστα ώστε να γίνει αύξηση ή να έχει εντοπιστεί ίσος αριθμός νέων τοπικών βέλτιστων για να γίνει μείωση.

[3] $|X| \times |D|$ είναι το πλήθος των ζευγών $< V, u >$ για κάθε CSP

Algorithm 6 `DTS (P: a CSP, tl: the tabu tenure, maxmoves: the number of moves allowed)`

1: $s \leftarrow$ random complete variable assignment
2: $nbmoves \leftarrow 0$
3: $tl \leftarrow$ random value between $10 - 30$
4: initialize randomly the tabu list
5: **while** `eval`$(s) > 0$ **and** $nbmoves < maxmoves$ **do**
6: choose the $< V, u' >$ that minimizes constraint violations, where V is a conflicting variable
7: update the tabu list
8: assign u' to V
9: $nbmoves \leftarrow nbmoves + 1$
10: **if** `eval`$(s) < bf$ **then**
11: $bf \leftarrow$ `eval`(s)
12: `delBestCanditates()`, deletes the list with the stored best solution candidates
13: `addtoBestCandidates`(s)
14: $forb \leftarrow 0$
15: **else if** `eval`$(s) = bf$ **then**
16: **if** `existinBestCandidates`(s) **then**
17: $forb \leftarrow forb + 1$
18: **if** $forb > 5tl$ **then**
19: `incrTL`(), increase tl by $\frac{|X| \times |D|}{100}$
20: **end if**
21: **else**
22: `addtoBestCandidates`(s)
23: $forb \leftarrow forb - 1$
24: **if** $forb = 0$ **then**
25: decrTL()
26: **end if**
27: **end if**
28: **end if**
29: **end while**
30: **return** s

Κεφάλαιο 4

Πειραματική Διαδικασία

4.1 Εισαγωγή

Οι αλγόριθμοι που υλοποιήθηκαν εξετάστηκαν ως προς την απόδοσή τους σε προβλήματα ικανοποίησης περιορισμών όπου οι περιορισμοί είναι δυαδικοί. Οι κλάσεις προβλημάτων που εξετάστηκαν ήταν από τη συλλογή του Christophe Lecoutre. Τα περισσότερα προβλήματα αυτής της συλλογής έχουν χρησιμοποιηθεί για τον έλεγχο της απόδοσης διάφορων ευριστικών μηχανισμών σε συνέδρια και διαγωνισμούς και έχουν αποτελέσει σημείο αναφοράς για τη σύγκριση αλγορίθμων βελτιστοποίησης.

4.2 Κλάσεις Προβλημάτων

Για τον πειραματικό έλεγχο χρησιμοποιήθηκαν διαφορετικές κλάσεις προβλημάτων με διαφορετική πυκνότητα λύσεων και δυσκολία. Παρακάτω γίνεται μια συνοπτική περιγραφή της κάθε κλάσης.

Composed

Αυτή η κλάση αποτελείται από τυχαία παραγόμενα CSPs αποτελούμενα από ένα κύριο κομμάτι και κάποια άλλα βοηθητικά κομμάτια τα οποία είναι «εμβολιασμένα» στο κύριο μέσω της εισαγωγής κάποιων δυαδικών περιορισμών. Μερικά από αυτά είναι συνεπή.

Driver

Τα προβλήματα αυτής της κλάσης είναι πραγματικά και έχουν γίνει σ΄ αυτά μερικές μικρές τροποποιήσεις από τους παρόχους τους ώστε να καταστούν ικανοποιήσιμα.

Geom

Το πρόβλημα Geometric έχει προταθεί από τον Rick Wallace. Τα στιγμιότυπα αυτά είναι τυχαία και δημιουργήθηκαν με τον εξής τρόπο: Αντί να δημιουργηθούν με βάση μια παράμετρο που ορίζει την πυκνότητα των λύσεων, χρησιμοποιήθηκε μία παράμετρος απόστασης dst τέτοια ώστε $dst < \sqrt{2}$. Για κάθε μεταβλητή επιλέχθηκαν τυχαία 2 συντεταγμένες ώστε το σχετικό σημείο τους να ανήκει σ΄ ένα μοναδιαίο τετράγωνο. Έπειτα, για κάθε ζεύγος μεταβλητών (x, y) αν η απόσταση των σχετικών τους σημείων είναι μικρότερη ή ίση της dst το τόξο (x, y) εισάγεται στον γράφο περιορισμών. Όλα τα στιγμιότυπα είναι ικανοποιήσιμα.

Hanoi

Πρόκειται για το γνωστό πρόβλημα των πύργων του Ανόι όπου στόχος είναι η μεταφορά ενός πύργου που αποτελείται από n δίσκους, ο ένας μεγαλύτερος από τον άλλο, από έναν πάσαλο σ΄ έναν άλλο με τον περιορισμό ότι κάθε φορά

μετακινείται ένας μόνο δίσκος και ότι ποτέ δεν τοποθετείται μεγαλύτερος δίσκος πάνω σε έναν μικρότερο. Κάθε στιγμιότυπο είναι ικανοποιήσιμο.

Langford

Η γενικευμένη εκδοχή αυτού του προβλήματος είναι να διαταχθούν k σύνολα αριθμών με τιμή από το 1 έως το n έτσι ώστε κάθε αριθμός m να απέχει m θέσεις από τον τελευταίο αριθμό.

QCP/QWH

Το Quasi-group Completion Problem έχει ως ζητούμενο τη διερεύνηση για το αν ένα μερικώς συμπληρωμένο Λατινικό Τετράγωνο μπορεί να συμπληρωθεί ώστε να αποτελέσει ένα πλήρες Λατινικό Τετράγωνο. Λατινικό Τετράγωνο τάξης n είναι ένας πίνακας $n \times n$ με ακριβώς n διαφορετικά σύμβολα όπου κάθε σύμβολο εμφανίζεται μια φορά σε μία γραμμή και μια φορά σε κάθε στήλη. Από τα QCP μερικά στιγμιότυπα είναι συνεπή ενώ τα υπόλοιπα όχι.

Τα Quasi-group With Holes είναι μια παραλλαγή των QCP και δημιουργήθηκαν έτσι ώστε να είναι όλα ικανοποιήσιμα. Αυτά τα στιγμιότυπα δημιουργήθηκαν από τον Radoslaw Szymanek για τον διαγωνισμό CSP Solver Competition του 2005.

Graph Coloring

Πρόκειται για την γνωστή κατηγορία προβλημάτων χρωματισμού γράφου. Για έναν δεδομένο γράφο $G = (E, V)$, όπου V τα τόξα και E οι κόμβοι το ζητούμενο είναι να βρεθεί ο ελάχιστος αριθμός k χρωμάτων ώστε κάθε κόμβος να λαμβάνει ένα χρώμα και το χρώμα κάθε γειτονικού κόμβου (συνδεδεμένων με τόξο) να είναι διαφορετικό. Ο ελάχιστος αριθμός k χρωμάτων που απαιτείται για την επίλυση του χρωματισμού ενός γράφου καλείται χρωματικός αριθμός.

Η αναγωγή του προβλήματος αυτού σε Πρόβλημα Απόφασης (Decision Problem) έχει ως στόχο την απάντηση στο ερώτημα για το αν ένας δεδομένος αριθμός k είναι χρωματικός αριθμός του γράφου G.

Οι υποκλάσεις που χρησιμοποιήθηκαν είναι οι εξής: fpsol, inithx, mug, mulsol, myciel, zeroin. Μερικές από αυτές είναι ικανοποιήσιμες.

4.3 Hardware / Software

Τα πειράματα εκτελέστηκαν σε υπολογιστή με επεξεργαστή Intel Quad Core 2.83GHz , 4GB RAM και λειτουργικό σύστημα Windows 7 x86. Οι αλγόριθμοι αναπτύχθηκαν στη γλώσσα προγραμματισμού C. Για την αυτοματοποιημένη εκτέλεση των προγραμμάτων αναπτύχθηκαν scripts τα οποία χρησιμοποιούσαν τους 3 από τους 4 πυρήνες του επεξεργαστή με κάθε πρόγραμμα να εκτελείται στον κάθε πυρήνα.

4.4 Κριτήρια Σύγκρισης

Από κάθε κλάση προβλημάτων επιλέχθηκαν στιγμιότυπα ικανοποιήσιμα και μη. Κάθε αλγόριθμος εκτελέστηκε 10 φορές για το κάθε στιγμιότυπο κάθε κλάσης με μέγιστο χρόνο τα 2 λεπτά. Για κάθε εκτέλεση καταγράφηκαν ως κριτήρια αξιολόγησης ο χρόνος εκτέλεσης σε δευτερόλεπτα, το ελάχιστο κόστος που έφτασε η αντικειμενική συνάρτηση καθώς και αριθμός των επαναλήψεων του αλγόριθμου.

Έπειτα, υπολογίστηκαν οι μέσοι όροι σε καθένα από τα παραπάνω κριτήρια για το κάθε στιγμιότυπο ξεχωριστά και στο τέλος υπολογίστηκε ο μέσος όρος των μέσων όρων της κάθε κλάσης προβλημάτων. Στους παρακάτω πίνακες παρατίθενται τα αποτελέσματα της κάθε κλάσης προβλημάτων για κάθε αλγόριθμο.

4.5 Ρύθμιση Παραμέτρων

Για τους αλγόριθμους MCRW, TS, LBS, SLBS και DTS χρειάστηκε να γίνει ρύθμιση των παραμέτρων τους. Οι Η ρύθμιση των παραμέτρων έγινε επιλέγοντας τυχαία στιγμιότυπα προβλημάτων και δοκιμάζοντας διαφορετικές τιμές από το επιτρεπτό εύρος τιμών της κάθε παραμέτρου για τα οποία έχουν γίνει προκαταρκτικές μελέτες. Τα προβλήματα αυτά δοκιμάστηκαν σε κάθε αλγόριθμο για περιορισμένο αριθμό επαναλήψεων και επιλέχθηκε για την καθεμιά η βέλτιστη τιμή.

MCRW

Στον MCRW επιλέχθηκε για την παράμετρο p η τιμή 0.1. Με άλλα λόγια, σε κάθε επανάληψη του αλγόριθμου, στην μεταβλητή που επιλέγεται προς τροποποίηση υπάρχει πιθανότητα 10% να της ανατεθεί μία τυχαία τιμή από το πεδίο τιμών της και 90% να εφαρμοστεί ο ευριστικός μηχανισμός MCRW. Η τιμή αυτή ορίστηκε στο πάνω άκρο του επιτρεπτού εύρους $0.02 < p < 0.1$ επειδή σε ορισμένα προβλήματα υψηλής δυσκολίας η ομαλότερη σύγκλιση προς τη λύση επιφέρει καλύτερα αποτελέσματα.

TS

Στην Αναζήτηση Ταμπού η παράμετρος tl καθορίζει το μέγεθος της λίστας ταμπού, δηλαδή πόσα ζεύγη μεταβλητής - τιμής $< V, u >$ αποτελούν απαγορευμένες κινήσεις κάθε χρονική στιγμή. Το επιτρεπτό εύρος τιμών γι΄ αυτήν την παράμετρο είναι το $10 < tl < 30$ και επιλέχθηκε γι΄ αυτή επίσης το άνω άκρο $tl = 30$ λόγω του ότι κάποια προβλήματα είχαν αρκετά μεγάλο πλήθος μεταβλητών άρα και μεγαλύτερη πιθανότητα να «εγκλωβίζουν» την αναζήτηση σε μία περιοχή γύρω από κάποιο τοπικό βέλτιστο. Στις περιπτώσεις όπου τα προβλήματα είχαν μικρό πλήθος μεταβλητών και πεδίων τιμών, το μέγεθος

30 της λίστας δεν έδειξε να επηρεάζει σημαντικά την απόδοση του αλγόριθμου ως προς το πλήθος των συγκρούσεων που έφτανε με δεδομένο το χρονικό περιθώριο των 2 λεπτών.

LBS

Στην Τοπική Ακτινική Αναζήτηση η παράμετρος που παίζει καθοριστικό ρόλο στην απόδοση είναι το πλήθος k των αρχικών καταστάσεων. Πειραματικά επιλέχθηκε η τιμή $k = 10$. Λόγω της πολύ υψηλής πολυπλοκότητας της μεθόδου, υψηλότερη τιμή του k οδηγούσε σε εκθετική επιβράδυνση της εκτέλεσης του αλγόριθμου αφού στην LBS εντοπίζονται όλες οι γειτονικές καταστάσεις σε κάθε μία από της k αρχικές, υπολογίζεται το πλήθος των συγκρούσεων και ταξινομούνται με βάση το κόστος της αντικειμενικής συνάρτησης της καθεμιάς. Μικρότερες τιμές του k μειώνουν αρκετά την απόδοση.

SLBS

Στη Στοχαστική Ακτινική Αναζήτηση η παράμετρος k ορίστηκε κι αυτή στην ίδια τιμή, $k = 10$. Η πολυπλοκότητα αυτής της μεθόδου αν και με καθορισμένο το ίδιο k είναι μεγαλύτερη αφού κάθε φορά επιλέγονται τυχαία k καταστάσεις από τη λίστα γειτνίασης με πιθανότητα ανάλογη με το πλήθος των συγκρούσεων που υπολογίζεται για την καθεμία.

DTS

Στην Δυναμική Αναζήτηση Ταμπού η παράμετρος tl ξεκινάει κάθε φορά από μία τυχαία τιμή στο εύρος $10 < k < 30$. Από κει και έπειτα η τιμή της, αρά και το μέγεθος της λίστας ταμπού, μεταβάλλεται αυτόματα με βάση τις συνθήκες της αναζήτησης που αναφέρονται στην περιγραφή του αλγόριθμου.

Το όριο ανοχής $forb$ στο πλήθος των επαναλαμβανόμενων επισκέψεων στα

ίδια τοπικά βέλτιστα το οποίο καθορίζει το αν πρέπει το μέγεθος της λίστας ταμπού να μεταβληθεί ή όχι μεταβάλλεται κι αυτό αυτόματα αφού είναι ανάλογο του μεγέθους της λίστας ταμπού τη κάθε χρονική στιγμή. Καθορίστηκαν εξ' αρχής το μέγιστο μέγεθος που μπορεί να λάβει η λίστα ταμπού από την συνάρτηση `incrTL()` το οποίο ορίστηκε στο 50% του πλήθους των ζευγών μεταβλητών - τιμών $< V, u >$ που ορίζονται από κάθε πρόβλημα καθώς και το μέγιστο πλήθος των στοιχείων της λίστας F των ήδη επισκεφθεισών τοπικών ελαχίστων στο πενταπλάσιο του μεγέθους της λίστας ταμπού.

4.6 Αποτελέσματα

Παρακάτω παρατίθενται οι πίνακες με τους μέσους όρους των καλύτερων λύσεων κάθε αλγόριθμου για το κάθε στιγμιότυπο προβλήματος και στο τέλος ο πίνακας με το πλήθος των στιγμιοτύπων που επιλύθηκαν σε κάθε κλάση προβλημάτων από τον κάθε αλγόριθμο.

Με βάση τα αποτελέσματα του πίνακα 4.1 παρατηρούμε ότι την καλύτερη απόδοση σε χρόνο εκτέλεσης 2 λεπτών την είχε ο DTS ο οποίος έφτασε κατά μέσο όρο τις 75.29 συγκρούσεις. Η διαφορά του με τον δεύτερο σε απόδοση, δηλαδή τον TS, όπως φαίνεται στα αποτελέσματα είναι στις 17.49 συγκρούσεις κάτι το οποίο αποτελεί ένδειξη για το ότι η δυναμική μεταβολή του μεγέθους της λίστας ταμπού καθιστά την αναζήτηση ταμπού περισσότερο αποδοτική. Επίσης παρατηρούμε ότι σε μικρά και εύκολα προβλήματα όπου όλοι οι αλγόριθμοι πλησίασαν πολύ κοντά στη λύση οι MC, MCRW, TS και DTS δεν παρουσίασαν μεγάλες διαφορές. Οι LBS και SLBS παρουσίασαν πολύ χειρότερη απόδοση από τους υπόλοιπους λόγω της υψηλής τους πολυπλοκότητας και τον σχετικά μικρό χρόνο εκτέλεσης.

Ο MCRW φαίνεται κατά μέσο όρο ελαφρώς χειρότερος από τον MC αλλά αυτό το διαμορφώνουν τα πολύ δύσκολα προβλήματα τα οποία εμφανίζουν

Πίνακας 4.1: Το μέσο κόστος των καλύτερων λύσεων για την κάθε κλάση προβλημάτων.

Problem Class	MC	MCRW	TS	LBS	SLBS	DTS
composed-25-1-2	3.25	2.0	2.04	3.57	3.53	2.0
composed-25-1-25	3.33	2.0	2.01	4.1	3.9	2.01
composed-75-1-25	3.56	2.0	2.81	6.5	6.08	2.02
composed-25-1-40	3.29	2.0	2.03	4.06	4.1	2.0
composed-25-1-80	3.81	2.0	2.0	4.34	4.41	2.0
composed-75-1-2	3.32	2.0	2.82	5.92	5.89	2.24
driverlogw	425.36	425.59	220.16	692.26	697.96	207.55
geom	6.67	6.51	1.18	9.46	7.58	0.31
hanoi	1.0	0.75	0.53	16.65	16.8	0.75
fpsol	568.81	582.11	325.31	682.32	691.14	331.15
langford	2.8	2.18	1.75	6.15	6.33	1.2
qcp-15	11.09	12.14	8.14	33.72	31.27	4.56
qcp-20	79.41	81.78	23.07	221.18	235.22	16.05
qwh-10	4.22	0.22	0.2	22.14	20.36	0.0
qwh-15	9.22	9.38	5.17	41.27	40.75	2.14
qwh-20	96.68	95.65	17.29	148.84	150.35	12.35
qwh-25	285.96	289.84	237.57	321.4	320.88	191.21
inithx	1345.15	1349.26	889.14	1396.7	1387.06	600.22
mug	1.94	0.5	0.5	12.67	10.08	0.5
mulsol	101.13	104.41	102.52	121.39	120.91	102.59
myciel	8.67	7.97	7.83	35.64	33.18	7.78
zeroin	87.59	91.21	90.52	108.74	109.0	90.5
TOTAL AVG	145.54	146.26	92.6	184.87	186.04	**75.29**

Πίνακας 4.2: Το πλήθος των επιλυμένων στιγμιοτύπων σε κάθε κλάση προβλημάτων

Problem Class	MC	MCRW	TS	LBS	SLBS	DTS
composed-25-1-2	0	0	0	0	0	0
composed-25-1-25	0	0	0	0	0	0
composed-75-1-25	0	0	0	0	0	0
composed-25-1-40	0	0	0	0	0	0
composed-25-1-80	0	0	0	0	0	0
composed-75-1-2	0	0	0	0	0	0
driverlogw	1	1	3	1	1	5
geom	0	0	8	0	0	9
hanoi	0	1	2	0	0	2
fpsol	0	0	0	0	0	0
langford	1	1	2	1	0	2
qcp-15	0	0	0	0	0	0
qcp-20	0	0	0	0	0	0
qwh-10	1	10	10	0	0	10
qwh-15	0	0	5	0	0	5
qwh-20	0	0	0	0	0	0
qwh-25	0	0	0	0	0	0
inithx	0	0	0	0	0	0
mug	4	4	4	0	0	4
mulsol	0	0	0	0	0	0
myciel	5	5	0	0	5	5
zeroin	0	0	0	0	0	0
TOTAL	12	22	34	2	6	**42**

μεγάλο πλήθος συγκρούσεων. Αφαιρώντας τα δύσκολα προβλήματα, δηλαδή τα ακραία σημεία της κατανομής στους MC και MCRW, παρατηρούμε ότι ο MCRW πλεονεκτεί έναντι του MC και αυτό μας οδηγεί στο συμπέρασμα ότι ο μέγιστος χρόνος εκτέλεσης δεν ήταν επαρκής για να αναδειχθεί η ικανότητα του MCRW να υπερβαίνει τα τοπικά βέλτιστα.

Στην σύγκριση των MCRW και TS παρατηρούμε ότι σε εύκολα προβλήματα παρουσιάζουν σχεδόν την ίδια απόδοση. Σε μερικές περιπτώσεις ο MCRW είναι κατά λίγο καλύτερος αλλά όσο αυξάνεται η δυσκολία των προβλημάτων η απόδοση του TS αυξάνεται εκθετικά συγκριτικά με τον MCRW καθιστώντας τον πρώτο καλύτερη επιλογή για την επίλυση προβλημάτων ικανοποίησης περιορισμών.

Στον πίνακα 4.2 παρατηρούμε πόσα στιγμιότυπα κατάφερε να επιλύσει ο κάθε ευριστικός μηχανισμός. Εδώ επιβεβαιώνεται ο ισχυρισμός μας για την καλύτερη απόδοση του MCRW συγκριτικά με τον MC. Ακόμη, με αυτά τα αποτελέσματα συμπεραίνουμε ότι ο SLBS έχει μεγαλύτερη πιθανότητα να οδηγηθεί σε λύση συγκριτικά με την μη στοχαστική του εκδοχή δηλαδή τον LBS. Τέλος, παρατηρούμε ότι ο DTS κατάφερε να λύσει τα περισσότερα προβλήματα με μεγάλη διαφορά από τους υπόλοιπους.

Κεφάλαιο 5

Σύνοψη / Συμπεράσματα

Στην παρούσα διπλωματική εργασία παρουσιάσαμε έξι διαφορετικούς αλγόριθμους για την επίλυση προβλημάτων ικανοποίησης περιορισμών. Αυτοί οι αλγόριθμοι εξετάστηκαν και συγκρίθηκαν σε προβλήματα διαβαθμισμένης δυσκολίας με διαφορετικό πλήθος μεταβλητών και περιορισμών. Η πρόταση μας για την δυναμική εκδοχή της Αναζήτησης Ταμπού εμφανίστηκε ιδιαίτερα ανταγωνιστική σε σχέση με τις καλύτερες κατά την βιβλιογραφία υπάρχουσες μεθόδους.

Η Δυναμική Αναζήτηση Ταμπού (Dynamic Tabu Search) με βάση τα αποτελέσματα των πειραμάτων είναι 23% καλύτερη σε απόδοση από την Αναζήτηση Ταμπού και τουλάχιστον 100% καλύτερη από τις υπόλοιπες μεθόδους που μελετήθηκαν και αυτό στα πλαίσια της ποιότητας των λύσεων όσον αφορά την ελαχιστοποίηση των συγκρούσεων. Με άλλα λόγια, οι λύσεις που κατάφερε να εντοπίσει ικανοποιούσαν το μεγαλύτερο πλήθος περιορισμών.

Με βάση τις κλάσεις προβλημάτων πάνω στις οποίες εξετάστηκαν οι αλγόριθμοι, ο DTS αναδείχθηκε εμφανώς ο καλύτερος σε απόδοση. Στην βιβλιογραφία αναφέρονται κι άλλα χαρακτηριστικά που μπορούν να βελτιώσουν την απόδοση της Αναζήτησης Ταμπού όπως η εντατικοποίηση (intensification) και η διαφοροποίηση (diversification) από τις οποίες η δεύτερη σε μια απλή μορφή της εφαρμόστηκε στην Δυναμική Αναζήτηση Ταμπού στα πλαίσια της μεταβο-

λής του μεγέθους της λίστας ταμπού κατά τον χρόνο εκτέλεσης της αναζήτησης κάτι το οποίο οδήγησε σε σημαντική βελτίωσή της.

Κεφάλαιο 6

Μελλοντική Εργασία

Τα ενθαρρυντικά αποτελέσματα της απόδοσης της DTS αποτέλεσαν κίνητρο για την περαιτέρω βελτίωση της μεθόδου. Πρόκειται να γίνουν δοκιμές στον αλγόριθμο με πιο σύνθετες συναρτήσεις γειτνίασης όπου ένα ποσοστό των γειτονικών καταστάσεων θα διαφοροποιείται από την γονική σε περισσότερες από μία τιμές μεταβλητών.

Ακόμη μελετώνται αυτή την στιγμή γενετικοί αλγόριθμοι οι οποίοι αποτελούν εξελιγμένη μορφή της τοπικής ακτινικής αναζήτησης στις οποίες γίνεται βελτιστοποίηση σε επιμέρους υποσύνολα των μεταβλητών ενός προβλήματος. Επίσης, πρόκειται να τροποποιηθούν οι αλγόριθμοι ώστε να εξεταστούν σε on-line προβλήματα κάτι το οποίο παρουσιάζει μεγάλο ερευνητικό ενδιαφέρον αφού είναι κάτι που βρίσκει εφαρμογή σε αληθινά προβλήματα στη βιομηχανία.

Βιβλιογραφία

[1] Mackworth, Alan K. Constraint Satisfaction. *Encyclopedia of Artificial*, 1985.

[2] Fahiem Bacchus and Xinguang Chen and Peter Van Beek and Toby Walsh. Binary vs. non-binary constraints. *Artificial Intelligence*, 140: 2002, 2002.

[3] Mark D. Johnston. Automated telescope scheduling. In *in Proc. Conf. on Coordination of Observational Projects*, page 219. Univ. Press, 1988.

[4] P. Galinier and J.K. Hao. Tabu Search for Maximal Constraint Satisfaction Problems. *CP*, pages 196–208, 1997.

[5] BT Smith, JM Boyle, JJ Dongarra, BS Garbow, Y Ikebe, VC Klema, CB Moler, and EISPACK Guide. Lecture notes in computer science. *Matrix eigensystem routines-EISPACK guideSpringer, Berlin*, 1976.

[6] Kostas Stergiou. Heuristics for dynamically adapting propagation in constraint satisfaction problems. *Ai Communications*, 22(3):125–141, 2009.

[7] Dionisios Pothos and Barry Richards. An empirical study of min-con ict hill climbing and weak commitment search. *Imperial College, London*, 1996.

[8] Nikolaos Samaras and Kostas Stergiou. Binary encodings of non-binary constraint satisfaction problems: Algorithms and experimental results. *Journal of Artificial Intelligence Research*, 24(1):641–684, 2005.

[9] Tomás Feder and Pavol Hell. Full constraint satisfaction problems. *SIAM Journal on Computing*, 36(1):230–246, 2006.

[10] Markus Bohlin. Constraint satisfaction by local search. *SICS Research Report*, 2002.

[11] Roberto Battiti and Giampietro Tecchiolli. The reactive tabu search. *ORSA journal on computing*, 6(2):126–140, 1994.

[12] E Taillard. Robust taboo search for the quadratic assignment problem. *Parallel computing*, 17(4):443–455, 1991.

[13] Said Salhi. Defining tabu list size and aspiration criterion within tabu search methods. *Computers & Operations Research*, 29(1):67–86, 2002.

[14] T Stützle. Local search algorithms for combinatorial problems. *Darmstadt University of Technology PhD Thesis*, 1998.

[15] Michel Gendreau, Gilbert Laporte, and Jean-Yves Potvin. Metaheuristics for the capacitated vrp. *The vehicle routing problem*, 9:129–154, 2002.

[16] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Englewood Cliffs, 1995.

[17] Christos H Papadimitriou. *Computational complexity*. John Wiley and Sons Ltd., 2003.