ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

[HRY 419]-Ανάπτυξη Εργαλείων CAD για Σχεδίαση Ολοκληρωμένων Κυκλωμάτων

Αναφορά 6ου εργαστηρίου

Ιωάννης Περίδης
A.M. 2018030069

28 Μαΐου 2022

1 Εισαγωγή:

Σκοπός του εργαστηρίου αυτού, είναι η κατανόηση της απόδοσης των ίδιων των εργαλείων cad που έχουμε δημιουργήσει ως τώρα και η σύγκρισή τους σε πολλούς διαφορετικούς τομείς. Συγκεκριμένα, θα συγκρίνουμε χρόνους εκτέλεσης , αριθμούς επαναλήψεων και αρκετά άλλα στατιστικά των αλγορίθμων tplace & route των εργαστηρίων 4,5 και 6. Οι συγκρίσεις αυτές φυσικά, δεν είναι ακριβώς βασισμένες στην πραγματικότητα, γιατί έχουμε δεχτεί διάφορες απλοποιήσεις σε σχέση με το μέγεθος των προγραμμάτων στις μνήμες, παρόλα αυτά είναι αρκετές για την ορθή αξιολόγηση των αλγορίθμων ,αλλά και της ταχύτητας των κωδίκων.

2 Αποτελέσματα:

Παρακάτω θα γίνει σύγκριση κάποιων μετρήσεων για κάθε ένα από τα εργαστήρια 4,5 και 6. Συγκεκριμένα θα αναλυθούν οι αλγόριθμοι για place & route, του Simulated Annealing, της θεωρίας γράφων και τέλος του συνδυασμού των 2 αυτών. Για να είναι δίκαια φυσικά η σύγκριση των αλγορίθμων , χρησιμοποιήθηκε το ίδιο αρχικά recipe αρχικά και για τους 2 (αυτούς που είχαν μέσα simulated annealing). Ύστερα, θα πειραματιστούμε με αυτά τα recipe, ξεχωριστά για κάθε αλγόριθμο. Τέλος, το αρχείο στο οποίο θα γίνουν οι μετρήσεις , είναι το Askhsh_2_FA_8bit_v2, το οποίο ,είναι ο 8bit αθροιστής , που αρχίζει από ένα τελείως τυχαίο placement.

<u>Το recipe αυτό είναι το εξής:</u>
Συνθήκη Τερματισμού: termination=100
Συνθήκη Αποδοχής Χειρότερης Λύσης: acceptBadSolution=8 και acceptBadSolution*2 , μετά από κάθε τέτοια αποδοχή.

Θα συγκριθούν τα εξής:

1) <u>Οι γραμμές στην τελική λύση</u>. Προφανώς, θέλουμε να είναι όσο το δυνατόν λιγότερες ,εφόσον είναι το κριτήριο της αποδοτικότητας της λύσης.

2) <u>Οι συνολικές επαναλήψεις που χρειάστηκαν για να φτάσουμε στην λύση αυτή</u>. Είναι ένας εξαιρετικά σημαντικός παράγοντας, αφού μας δείχνει σε έναν βαθμό την ταχύτητα της λύσης (φυσικά δεν εξαρτάται μόνο από αυτό). Είναι φανερό όμως, πως αν βρούμε μια λύση με πολύ λίγες γραμμές, αλλά χρειάζεται πάρα πολλές επαναλήψεις για να συγκλίνει, τότε πιθανότατα ανάλογα σε τι κύκλωμα θα

χρησιμοποιήσουμε τον αλγόριθμο ,δεν θα είναι χρήσιμός , λόγω χρονικών περιορισμών που θα έχουμε. Ακόμη, πρέπει να γίνει ξεκάθαρο , πώς όταν αναφερόμαστε στον όρο επαναλήψεις, εννοείται το πλήθος των φορών που χρειάστηκε να τρέξει ο αλγόριθμος RouteAndCompact().

3) Τα συνολικά δευτερόλεπτα που χρειάστηκαν για να τρέξει το πρόγραμμα. Είναι ένας ακόμα παράγοντας που μας δίνει την ταχύτητα σε πιο ρεαλιστικό επίπεδο. Φυσικά σχετίζονται (είναι ανάλογα) με τον αριθμό επαναλήψεων, αλλά δίνουν μια συνολική εκτίμηση της ταχύτητας , για όλα τα κομμάτια της υλοποίησης του αλγορίθμου και όχι μόνο για τις επαναλήψεις. Να σημειωθεί πως ο χρόνος αυτός είναι ο ολικός χρόνος που δίνεται από την C αφού τρέξεις ένα πρόγραμμα. Επομένως, εμπεριέχει και τα I/O , για το διάβασμα του αρχείου εισόδου, γέμισμα των πινάκων με τις τιμές που διαβάστηκαν και την εγγραφή του αρχείου εξόδου. Αυτοί οι χρόνοι, εκτός του ότι είναι πολύ μικροί σε σχέση με το συνολικό χρόνο του προγράμματος, είναι επίσης όλοι ίδιοι σε όλα τα εργαστήρια ,άρα στην σύγκριση τους, δεν έχουν κάποιο αντίκτυπο. Τέλος, οι χρόνοι αυτοί , ίσως είναι αρκετά διαφορετικοί από όταν τρέξετε ξανά τα εργαστήρια εσείς, γιατί καταμετρήθηκαν αφού έβαλα σε σχόλια όλες τις εκτυπώσεις πού ήταν μη αναγκαίες για την λειτουργία του προγράμματος (αλλά υπήρχαν εκεί για την δική σας πιο εύκολη κατανόηση των αποτελεσμάτων και την απεικόνιση τους στις αναφορές).

4) Ο αριθμός της επανάληψης στην οποία έγινε η τελευταία αποδοχή καλύτερης λύσης. Το στατιστικό αυτό, αρχικά φαίνεται να μην μας ενδιαφέρει, όμως αντιθέτως ,μπορεί να αποδειχθεί αρκετά χρήσιμο. Σε ένα ιδανικό σενάριο, που θα είχαμε βρει την βέλτιστη συνταγή για τους αλγορίθμους, τότε ο αριθμός της τελευταίας καλύτερης αποδοχής θα ταυτιζόταν με τον αριθμό ολικών επαναλήψεων. Αυτό διότι προφανώς, οι επαναλήψεις που κάνει ο αλγόριθμος όσο προχωράει και χωρίς να προλάβει να αποδεχτεί κάποια καλύτερη λύση πριν τερματίσει, είναι ανούσιες, αφού η τελευταία αλλαγή που θα γίνει θα είναι στην τελευταία αποδοχή καλύτερης λύσης. Αποτέλεσμα αυτού, είναι να γίνονται πολλές περιττές επαναλήψεις και να αυξάνεται ο χρόνος εκτέλεσης του προγράμματος. Συμπεραίνουμε λοιπόν, πώς όσο μεγαλύτερη είναι η διαφορά μεταξύ των ολικών επαναλήψεων και της τελευταίας καλύτερης αποδοχής, τόσο πιο μεγάλη δυνατότητα έχουμε , για να βελτιώσουμε την ταχύτητα του αλγορίθμου (μειώνοντας επαναλήψεις), απλά πειράζοντας την τερματική συνθήκη της συνταγής.

5) Το μήκος του μεγαλύτερου σύρματος και ο μέσος όρος μήκους όλων των συρμάτων. Προφανώς, θέλουμε οι αριθμοί αυτοί να είναι όσο το δυνατόν μικρότεροι , αφού σκοπός μας, είναι να χωρέσουμε το κύκλωμα στον ελάχιστο δυνατό χώρο. Το μήκος των συρμάτων, είναι ανάλογο των γραμμών στην λύση, όσο μικρότερο μήκος έχουν τα σύρματα, τότε πιθανότατα ,τόσο λιγότερες θα είναι και οι γραμμές. Να σημειωθεί πως το μήκος αυτό , δεν είναι κανονικό μήκος σε κάποια μονάδα μέτρησης, αλλά μετριέται σε απόσταση standard shells.

6) Το πλήθος των καλύτερων λύσεων που αποδεχτήκαμε και το πλήθος των χειρότερων λύσεων που αποδεχτήκαμε. Πάντα θέλουμε να υπάρχει μια ισορροπία μεταξύ των καλών και των κακών λύσεων που αποδεχόμαστε , καθώς και δεν θέλουμε να κολλήσουμε σε κάποιο τοπικό ελάχιστο σειρών, αλλά ούτε να αποδεχόμαστε και πολλές χειρότερες λύσεις.

7) Ο βέλτιστος σπόρος που βρέθηκε .Είναι αναμενόμενο, πως οι αλγόριθμοι θα δίνουν την βέλτιστή λύση τους για διαφορετικούς σπόρους (σαν είσοδο στην rand()). Για να γίνει δίκαια λοιπόν η σύγκρισή τους, χρησιμοποιήθηκε στην καταγραφή των αποτελεσμάτων ο καλύτερος δυνατός σπόρος για κάθε αλγόριθμο και όχι ο ίδιος και για τους 2.

Παρακάτω φαίνεται ο πίνακας με όλα τα μετρικά που αναφέρθηκαν παραπάνω:

Algorithm used for Place & Route	*Rows in Solution*	*Total Iterations*	*Total Seconds*	*Iteration Number of Last Better Solution Accepted*	*Biggest Wire Length*	*Mean Length of All Wires*	*Total Better Solutions Accepted*	*Total Worse Solutions Accepted*	*Best Seed*
Simulated Annealing (lab 4)	11	337	0,329 (329 ms)	236	33	4,65	7	4	6
Graph Theory (lab 5)	14	24	0,096 (96 ms)	-	28	4,65	-	-	-
Graph Theory & Simulated Annealing (lab 6)	11	24+288=312	0,251 (251 ms)	187	31	4,52	4	4	10

Παρακάτω φαίνονται τα βήματα του αλγορίθμου Simulated Annealing του 8 Bit μετρητή:

```
***These are the steps of the Place & Route algorithm:***
=========================================================
We found a better solution: Iteration Number:  1 | Total Iterations:  0 | Rows in Solution:19
We found a better solution: Iteration Number:  1 | Total Iterations:  1 | Rows in Solution:17
We found a better solution: Iteration Number:  1 | Total Iterations:  2 | Rows in Solution:16
We accept a worse solution: Iteration Number:  9 | Total Iterations: 10 | Rows in Solution:20
We accept a worse solution: Iteration Number: 17 | Total Iterations: 26 | Rows in Solution:18
We accept a worse solution: Iteration Number: 33 | Total Iterations: 58 | Rows in Solution:21
We found a better solution: Iteration Number: 54 | Total Iterations:112 | Rows in Solution:15
We accept a worse solution: Iteration Number: 65 | Total Iterations:176 | Rows in Solution:21
We found a better solution: Iteration Number: 56 | Total Iterations:232 | Rows in Solution:14
We found a better solution: Iteration Number:  3 | Total Iterations:235 | Rows in Solution:12
We found a better solution: Iteration Number:  1 | Total Iterations:236 | Rows in Solution:11

*****ROWS IN FINAL SOLUTION*****
=========================================
Number of rows after the Place & Route algorithm:11
The total Iterations needed:337
```

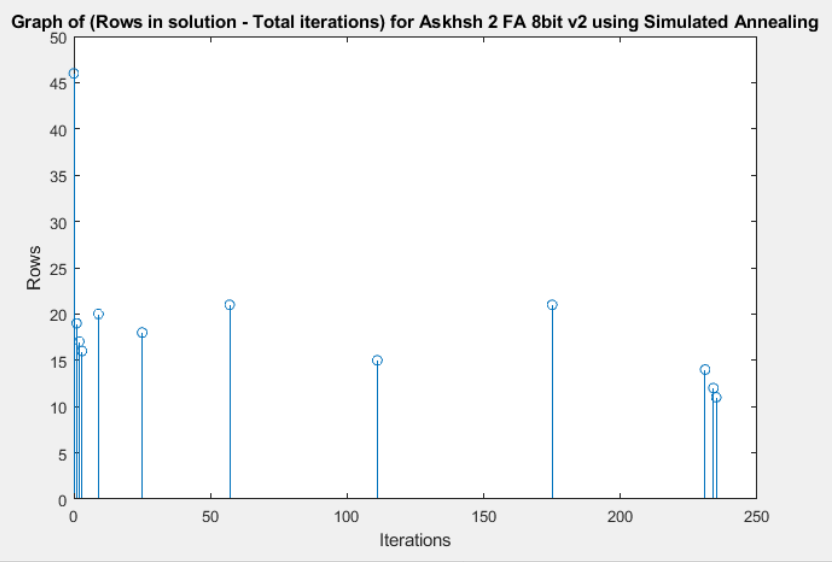
Παρακάτω φαίνεται το αποτέλεσμα του αλγορίθμου Graph-Theory του 8 Bit μετρητή, μετά τις μεταρρυθμίσεις του placement :

```
*****ROWS IN FINAL SOLUTION*****
=========================================
Number of rows after the Place & Route algorithm:14
```

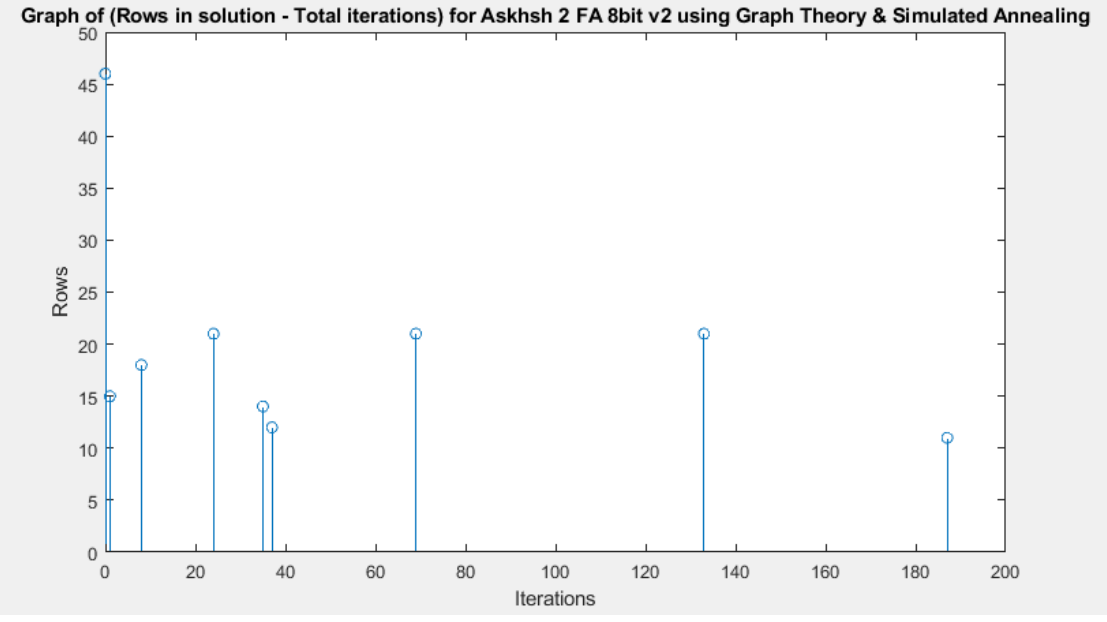
Παρακάτω φαίνονται τα βήματα του αλγορίθμου Graph-Theory +Simulated Annealing του 8 Bit μετρητή:

```
***These are the steps of the Place & Route algorithm:***
============================================================
We found a better solution: Iteration Number:  1 | Total Iterations:  0 | Rows in Solution:15
We accept a worse solution: Iteration Number:  9 | Total Iterations:  8 | Rows in Solution:18
We accept a worse solution: Iteration Number: 17 | Total Iterations: 24 | Rows in Solution:21
We found a better solution: Iteration Number: 11 | Total Iterations: 35 | Rows in Solution:14
We found a better solution: Iteration Number:  2 | Total Iterations: 37 | Rows in Solution:13
We accept a worse solution: Iteration Number: 33 | Total Iterations: 69 | Rows in Solution:21
We accept a worse solution: Iteration Number: 65 | Total Iterations:133 | Rows in Solution:21
We found a better solution: Iteration Number: 54 | Total Iterations:187 | Rows in Solution:11

*****ROWS IN FINAL SOLUTION*****
========================================
Number of rows after the Place & Route algorithm:11
The total Iterations needed:288
```

Παρακάτω φαίνεται η γραφική παράσταση (Rows in solution-Total iterations) του αλγορίθμου Simulated Annealing:



Παρακάτω φαίνεται η γραφική παράσταση (Rows in solution-Total iterations) του αλγορίθμου Graph-Theory +Simulated Annealing:



Αν ενώσουμε τα σημεία των γραφικών παραστάσεων αυτών , μόνο των λύσεων , θα πάρουμε μια φθίνουσα γραφική παράσταση που θα μοιάζει με την μορφή εκφόρτωσης ενός πυκνωτή. Αντίστοιχα, αν σχεδιάσουμε μια γραμμή στο αριθμητικό μέσο όλων των αποδεκτών λύσεων σε κάθε σημείο, πάλι θα απεικονιστεί κάτι παρόμοιο (ίσως με λίγο μεγαλύτερη απόκλιση σε κάποια σημεία).

3 Συγκρίσεις:

Όλες οι συγκρίσεις μας, θα γίνουν πάνω στο αρχικό recipe (όπου αυτό χρησιμοποιείται).

- ***Ξεκινώντας, θα συγκρίνουμε την σκέτη μέθοδο Simulated Annealing με την σκέτη μέθοδο της θεωρίας των γράφων.***

Algorithm used for Place & Route	***Rows in Solution***	***Total Iterations***	***Total Seconds***
Simulated Annealing (lab 4)	Καλύτερος	Χειρότερος κατά 1034,1% (313 επαναλήψεις παραπάνω)	Χειρότερος κατά 242,7% (233 ms παραπάνω)
Graph Theory (lab 5)	Χειρότερος κατά 21,4% (3 σειρές παραπάνω)	Καλύτερος	Καλύτερος

Ακόμη, ο μέσος όρος μήκους όλων των καλωδίων είναι ο ίδιος.

Παρατηρώ , όπως και ήταν αναμενόμενο με χρήση της θεωρίας των γράφων, οι συνολικές επαναλήψεις είναι εξαιρετικά λιγότερες (αυτό διότι για τον γράφο , χρειαζόμαστε μονάχα 24 επαναλήψεις του RouteAndCompact(), μόνο και μόνο για να βρούμε τον βέλτιστο τρόπο τοποθέτησης των υπογράφων στον πίνακα του Placement). Αυτό , έχει ως αποτέλεσμα, να έχει και αρκετά μικρότερο χρόνο εκτέλεσης , άρα να είναι ταχύτερο. Παρόλα αυτά, δεν δίνει το ίδιο καλή λύση, καθώς και είναι απλά , ένας εξυπνότερος τρόπος για να τοποθετηθεί το πρώτο placement και δεν αποτελεί από μόνο του , έναν πλήρη αλγόριθμο place & route, παρά ένα καλό αρχικό βήμα.

- ***Συνεχίζοντας , θα πάμε να συγκρίνουμε την μέθοδο Simulated Annealing με την μέθοδο της θεωρίας των γράφων, συνδυασμένη με την Simulated Annealing.***

Algorithm used for Place & Route	***Rows in Solution***	***Total Iterations***	***Total Seconds***	***Biggest Wire Length***	***Mean Length of All Wires***
Simulated Annealing (lab 4)	Ίδιο	Χειρότερος κατά 8.0% (25 επαναλήψεις παραπάνω)	Χειρότερος κατά 31,0% (78 ms παραπάνω)	Χειρότερος κατά 6,4% (2 μονάδες μήκους παραπάνω)	Χειρότερος κατά 2,8% (0,13 μονάδες μήκους παραπάνω)
Graph Theory & Simulated Annealing (lab 6)	Ίδιο	Καλύτερος	Καλύτερος	Καλύτερος	Καλύτερος

Παρατηρώ πως οι 2 αλγόριθμοι δίνουν το ίδιο αποτέλεσμα σε γραμμές, αλλά διαφέρουν στην ταχύτητά τους, με τον συνδυασμό των 2 μεθόδων να είναι η πιο αποδοτική λύση από όλες. Γεγονός που ήταν αναμενόμενο, καθώς και με την χρήση θεωρίας γράφων , γλιτώνουμε 3 αποδοχές καλύτερης λύσης (από 7 σε 4), καθώς και αρχίζουμε ήδη από μια καλύτερη λύση. Εφόσον λοιπόν, καταφέραμε να φτάσουμε στην καλύτερη αρχική αυτή λύση με μονάχα 24 επαναλήψεις, που είναι μικρότερο νούμερο , από τις επαναλήψεις που χρειάστηκε ο απλός Simulated Annealing, για να φτάσει στην ίδια κατάσταση με εμάς , τότε είναι φανερό πως γλιτώσαμε κάμποσες επαναλήψεις και άρα βελτιώσαμε την ταχύτητα (λιγότερος χρόνος).

Επίσης, παρατηρούμε και μια μικρή μείωση στο μέσο μήκος των συρμάτων.
Να σημειωθεί πως στο εργαστήριο 5, έχει γίνει ένα μικρό λαθάκι και το μέσο μήκος φαίνεται ίσο με 4 και όχι 4,52, γιατί είχα ξεχάσει να το κάνω float και ήταν int

Ακόμη, η διαφορά (*total iterations - iteration of last better solution accepted)* ,του lab 4 , είναι ίση με 101, ενώ του lab 6, ίση με 125. Άρα και πάλι είναι καλύτερο το lab 6, γιατί αυτό σημαίνει πως έχει μεγαλύτερο περιθώριο βελτίωσης της ταχύτητας, μικραίνοντας περισσότερο την τερματική συνθήκη του αλγορίθμου.

4 Αλλαγή Recipe:

Θα δοκιμάσουμε να κάνουμε κάποιες αλλαγές στα recipe, με σκοπό να πετύχουμε την ίδια λύση σε αριθμό σειρών, αλλά με λιγότερες επαναλήψεις και άρα μεγαλύτερη ταχύτητα.

Αρχικά, δοκιμάζουμε να κρατήσουμε σταθερό το όριο αποδοχής χειρότερης λύσης στο 8 και μειώνουμε τον αριθμό τερματικής συνθήκης, από 100 ,όσο περισσότερο γίνεται, χωρίς να αυξηθεί ο αριθμός των σειρών πάνω από 11. Με την διαδικασία αυτή, καταφέρνουμε να μικρύνουμε το κενό μεταξύ τελευταίας αποδοχής και ολικών επαναλήψεων, επομένως, θα μειώσουμε όσο το δυνατόν περισσότερο τις άσκοπες επαναλήψεις που γίνονται μετά την τελευταία αποδοχή λύσης. Το νέο όριο λοιπόν αυτό , μέχρι το οποίο δεν επηρεάζεται η ποιότητα της λύσης είναι **terminal=64.**

Παρακάτω φαίνονται τα αποτελέσματα για το νέο recipe με Terminal=64 και AcceptBadSolution=8:

Algorithm used for Place & Route	***Rows in Solution***	***Total Iterations***	***Total Seconds***
Simulated Annealing (lab 4)	11	301 (Καλύτερος κατά 11,9% , 36 επαναλήψεις λιγότερες από αρχικό recipe)	0,268 (268 ms) (Καλύτερος κατά 22,7% , 61 ms λιγότερα από αρχικό recipe)
Graph Theory & Simulated Annealing (lab 6)	11	252 (Καλύτερος κατά 23,8% 60 επαναλήψεις λιγότερες από αρχικό recipe)	0,212 (212 ms) (Καλύτερος κατά 18,3% , 39 ms λιγότερα από αρχικό recipe)

Όπως παρατηρείται , πράγματι βελτιώθηκε σε ένα αρκετά ικανοποιητικό ποσοστό η ταχύτητα και οι επαναλήψεις που χρειάζονται για να συγκλίνουμε στην τελική λύση, χωρίς να χρειαστεί να αλλάξουμε τίποτα στους αλγόριθμούς μας και στους κώδικες, απλά βρίσκοντας με δοκιμές ένα καλύτερο recipe.

Παρακάτω φαίνονται τα αποτελέσματα του προγράμματος για το νέο recipe, για τα 2 διαφορετικά εργαστήρια :

```
*****ROWS IN FINAL SOLUTION*****
========================================
Number of rows after the Place & Route algorithm:11
The total Iterations needed:301

Process returned 0 (0x0)   execution time : 0.268 s
```

```
*****ROWS IN FINAL SOLUTION*****
========================================
Number of rows after the Place & Route algorithm:11
The total Iterations needed:252

Process returned 0 (0x0)   execution time : 0.212 s
```

Συνεχίζοντας, θα δοκιμάσουμε να αλλάξουμε πλέον και το όριο αποδοχής κακής λύσης. Θα το αυξήσουμε κατά έναν πολύ μικρό αριθμό, με σκοπό να είμαστε λίγο πιο άπληστοι και να αποδεχόμαστε κακές λύσεις πιο σπάνια. Αυτό φυσικά σημαίνει πως πιθανότατα θα χρειαστούμε παραπάνω επαναλήψεις από 64 που είχαμε πριν , καθώς και αν τις κρατήσουμε ίδιες μπορεί να κολλήσουμε σε κάποιο τοπικό ελάχιστο γραμμών που να είναι χειρότερο από την πρώτη συνταγή. Τα νέα όρια λοιπόν, μετά από δοκιμές είναι **AcceptBadSolution=9 και Terminal=74.**

Παρακάτω φαίνονται τα αποτελέσματα για το νέο recipe με Terminal=74 και AcceptBadSolution=9:

Algorithm used for Place & Route	***Rows in Solution***	***Total Iterations***	***Total Seconds***
Simulated Annealing (lab 4)	11	311 (Καλύτερος κατά 8,3% , 26 επαναλήψεις λιγότερες από αρχικό recipe)	0,278 (278 ms) (Καλύτερος κατά 18,3% , 51 ms λιγότερα από αρχικό recipe)
Graph Theory & Simulated Annealing (lab 6)	11	262 (Καλύτερος κατά 19,0% 50 επαναλήψεις λιγότερες από αρχικό recipe)	0,214 (214 ms) (Καλύτερος κατά 17,2% , 37 ms λιγότερα από αρχικό recipe)

Αντίστοιχες παρατηρήσεις ισχύουν και για αυτά τα αποτελέσματα, απλά σε λίγο μικρότερα ποσοστά.

Παρακάτω φαίνονται τα αποτελέσματα του προγράμματος για το νέο recipe, για τα 2 διαφορετικά εργαστήρια :

```
*****ROWS IN FINAL SOLUTION*****
========================================
Number of rows after the Place & Route algorithm:11
The total Iterations needed:311

Process returned 0 (0x0)   execution time : 0.278 s
```

```
*****ROWS IN FINAL SOLUTION*****
==========================================
Number of rows after the Place & Route algorithm:11
The total Iterations needed:262

Process returned 0 (0x0)   execution time : 0.214 s
```

5 Άπληστη Προσέγγιση :

Στο σημείο αυτό, θα προσπαθήσουμε να κατανοήσουμε με ένα παράδειγμα την σχέση ποιότητας λύσης και χρόνου εκτέλεσης. Πολλές φορές από το πρόγραμμά μας , επιθυμούμε να έχουμε διαφορετικές προτεραιότητες. Συγκεκριμένα, μπορεί να δίνεται είτε έμφαση στον περιορισμό του χρόνου, δηλαδή να πάρουμε κάποια λύση που είναι καλή , αλλά όχι η καλύτερη που μπορούμε να βρούμε ,αλλά να την έχουμε σε ένα συγκεκριμένο μικρό χρονικό διάστημα. Είτε μπορούμε να δώσουμε έμφαση στην ποιότητα της λύσης και να προσπαθήσουμε να βρούμε την βέλτιστη δυνατή λύση, όσο χρόνο και αν χρειαστεί. Παρακάτω , ακολουθήθηκε μια greedy εκδοχή του Simulated Annealing, όπου θέτουμε σε ένα τεράστιο νούμερο (που δεν θα φτάσουμε ποτέ) , την αποδοχή χειρότερης λύσης και επίσης, θέτουμε μια πολύ μεγάλη συνθήκη τερματισμού. Σκοπός αυτού, είναι να τρέξει για πολύ ώρα ο αλγόριθμος, δίνοντας μόνο καλύτερες λύσεις, μέχρι να φτάσουμε στην καλύτερη δυνατή. Να σημειωθεί πως μιλώντας πλέον για τόσο μεγάλα χρονικά όρια, δεν έχει σημασία πλέον αν θα χρησιμοποιήσουμε και θεωρία γράφων ή όχι.

Παρακάτω φαίνεται ο πίνακας που δείχνει τον αριθμό των επαναλήψεων σε σχέση με τον αριθμό γραμμών της λύσης για την μέθοδο Simulated Annealing :

Algorithm used for Place & Route	*Rows in Solution*	*Total Iterations*
Simulated Annealing (lab 4)	11	236
	10	3.670
	9	26.565
	8	40.710
	7	290.925

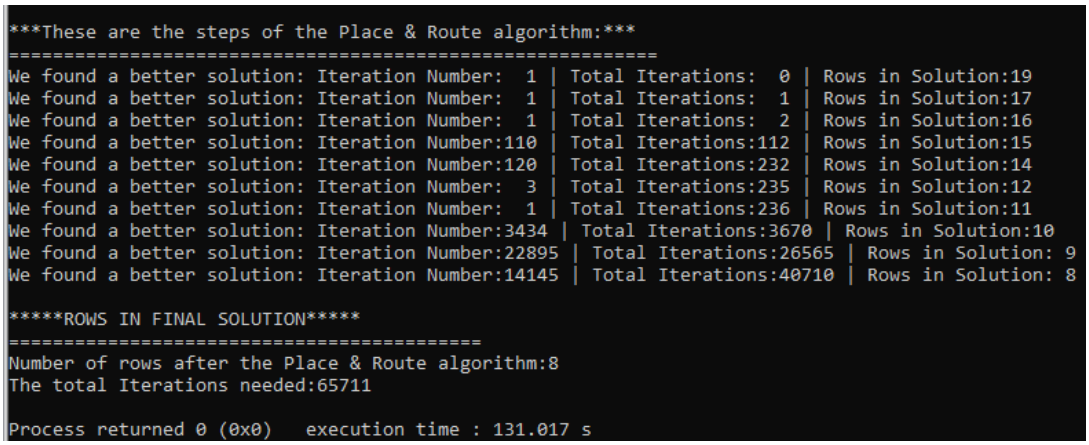
```
***These are the steps of the Place & Route algorithm:***
============================================================
We found a better solution: Iteration Number:  1 | Total Iterations:  0 | Rows in Solution:19
We found a better solution: Iteration Number:  1 | Total Iterations:  1 | Rows in Solution:17
We found a better solution: Iteration Number:  1 | Total Iterations:  2 | Rows in Solution:16
We found a better solution: Iteration Number:110 | Total Iterations:112 | Rows in Solution:15
We found a better solution: Iteration Number:120 | Total Iterations:232 | Rows in Solution:14
We found a better solution: Iteration Number:  3 | Total Iterations:235 | Rows in Solution:12
We found a better solution: Iteration Number:  1 | Total Iterations:236 | Rows in Solution:11
We found a better solution: Iteration Number:3434 | Total Iterations:3670 | Rows in Solution:10
We found a better solution: Iteration Number:22895 | Total Iterations:26565 | Rows in Solution: 9
We found a better solution: Iteration Number:14145 | Total Iterations:40710 | Rows in Solution: 8
We found a better solution: Iteration Number:250215 | Total Iterations:290925 | Rows in Solution: 7

*****ROWS IN FINAL SOLUTION*****
========================================
Number of rows after the Place & Route algorithm:7
```

Παρατηρείται λοιπόν, πως αν θέλουμε να πετύχουμε μικρότερο αριθμό σειρών , μπορούμε να το πετύχουμε , αλλά με μεγάλο χρονικό τίμημα. Όπως είναι φανερό, ο αριθμός επαναλήψεων που χρειάζονται για να μειωθεί κατά 1 μονάδα ο αριθμός σειρών στην λύση, αυξάνεται εξαιρετικά ραγδαία. Δηλαδή το κόστος σε επαναλήψεις για να περάσουμε από την σειρά 11 στην 10 , από την 10 στην 9 , από την 9 στην 8 και από την 8 στην 7, είναι κάθε φορά μεγαλύτερο από το άθροισμα όλων των υπολοίπων μαζί (και εξαιρετικά μεγαλύτερο ,όσο πηγαίνουμε βαθύτερα στον αλγόριθμο).

Παρακάτω φαίνονται τα αποτελέσματα για το νέο recipe με Terminal=23.000 και AcceptBadSolution= πρακτικά άπειρο:

Algorithm used for Place & Route	***Rows in Solution***	***Total Iterations***	***Total Seconds***	***Iteration Number of Last Better Solution Accepted***
Simulated Annealing (lab 4)	8	65.711	131,017 seconds	40.710

Όσον αφορά , το κόστος αυτό σε πραγματικό χρόνο, έχουμε ένα παράδειγμα για να φτάσουμε στις 8 γραμμές, χρειαζόμαστε 131 δευτερόλεπτα(!) , ενώ πριν ο χρόνος μετρούταν σε mili seconds , μόνο και μόνο για να μειώσουμε τις γραμμές κατά 3 , από 11 σε 8.
Οπότε γίνεται πλέον κατανοητό, πως το ποια λύση θα χρησιμοποιήσουμε, εξαρτάται πάντα από το ήδος του προβλήματος και τους περιορισμούς που έχει ο κατασκευαστής σε μέγεθος (δηλαδή σε πλήθος γραμμών που πρέπει να χωρέσει το κύκλωμα) και σε χρονικό περιθώριο που πρέπει να βγάζει το αποτέλεσμα.



6 Συμπέρασμα:

Συμπερασματικά, αποδείχθηκε ότι ο συνδυασμός μιας έξυπνης και στοχευμένης μετάθεσης των πυλών μέσω της θεωρίας γράφων, αντί τυχαίας , πάνω στο αρχικό Placement και ύστερα η υλοποίηση ενός πιθανοτικού αλγορίθμου , όπως το Simulated Annealing, δίνει την καλύτερη ποιότητα λύσης σε σχέση με τον χρόνο εκτέλεσης. Ακόμη, είναι φανερό πως όσο καλός και αν είναι ο αλγόριθμος, αν δεν βρεθεί μια καλή συνταγή για την χρήση του, τότε μπορεί τα αποτελέσματα να διαφέρουν εσθήτα από την μια λύση στην άλλη. Τέλος, είναι κατανοητό , πως δεν μπορούμε να πετύχουμε ένα βέλτιστο recipe για όλα τα κυκλώματα, ή ένα βέλτιστο seed ,ή έναν αλγόριθμο που θα δίνει πάντα τις καλύτερες λύσεις, αλλά μπορούμε με χρήση τεχνογνωσία , χρήση της θεωρίας και πολλές δοκιμές να βρίσκουμε ποιο εργαλείο ταιριάζει καλύτερα σε κάθε ένα κύκλωμα.