
Εθνικο Μετσοβιο Πολυτεχνειο

**ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

**ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ**

**ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ**


---

# ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

---

**3η ΑΣΚΗΣΗ**


*Αχλάτης Στέφανος-Σταμάτης (03116149)*

*<el16149@central.ntua.gr>*


Ιούνιος 2020

**Εισαγωγή**

Η παρούσα άσκηση έχει ως αντικείμενο τη διερεύνηση των πολυνηματικών επεξεργαστών και των χαρακτηριστικών τους, καθώς επίσης και τη μεταβολή της επίδοσης, της κατανάλωσης ενέργειας, και του μεγέθους του τσιπ τους. Στην πειραματική αξιολόγηση καλούμαστε να μελετήσουμε τα παραπάνω χαρακτηριστικά για διαφορετικά dispatch_width και window_size.

Πιο συγκεκριμένα, αρχικά θα μελετήσουμε την απόδοση των μετροπρογραμμάτων (Instructions per Cycle) για 72 διαφορετικους συνδυασμούς dispatch_width και window_size. Στη συνέχεια, θα ασχοληθούμε με την κατανάλωση ενέργειας για τους παραπάνω συνδυασμούς, αλλά και με το μέγεθος του chip τους. Τέλος με βάση τις παραπάνω προσωμοιώσεις και μετρήσεις θα εξάγουμε κάποια συμπεράσματα, συγκρίνοντας τις διαφορετικές υλοποιήσεις των παραμέτρων και αναδεικνύοντας τις βέλτιστες.

**Εργαλεία**

**PINPLAY**

Για την πραγματοποίηση της άσκησης χρησιμοποιήθηκε το εργαλείο PIN, το οποίο είναι ένα εργαλείο ανάλυσης εφαρμογών που αναπτύσσεται και συντηρείται από την Intel. Η χρήση του PINPLAY δίνει τη δυνατότητα για dynamic binary instrumentation, το οποίο σημαίνει πως εισάγεται δυναμικά κώδικας κατά τη διάρκεια της εκτέλεσης των μετροπρογραμμάτων ανάμεσα στις εντολές της εφαρμογής αυτής έτσι ώστε να συλλεχθούν πληροφορίες σχετικές με την εκτέλεση, όπως ο συνολικός αριθμός εντολών, ο συνολικός αριθμός ευστοχιών στην κρυφή μνήμη κ.ο.κ.

Η έκδοση του PIN στην οποία πραγματοποιήθηκε η προσομοίωση είναι η PINPLAY 3.11, σε Ubuntu Linux 18.04 με έκδοση πυρήνα 4.4.

**Sniper Multicore Simulator**

Θα χρησιμοποιήσουμε τον προσομοιωτή “Sniper Multicore Simulator”, ο οποίος αξιοποιεί το εργαλείο “PIN” που αναφέρεται παραπάνω. Με τη βοήθεια του Sniper Multicore Simulator θα μελετηθούν τα χαρακτηριστικά των σύγχρονων superscalar, out-of- order επεξεργαστών και ο τρόπος με τον οποίο επηρεάζουν την απόδοση του συστήματος, την κατανάλωση ενέργειας και το μέγεθος του chip του επεξεργαστή. Η έκδοση του Sniper που χρησιμοποιήθηκε είναι η Sniper7.3.

**McPAT**

Το McPAT (Multi-core Power, Area, Timing) είναι ένα εργαλείο το οποίο χρησιμεύει στη μοντελοποίηση των χαρακτηριστικών ενός επεξεργαστή, όπως για παράδειγμα η κατανάλωση ενέργειας και το μέγεθος που καταλαμβάνουν στο τσιπ οι διαφορετικές δομικές μονάδες του επεξεργαστή. Το McPAT συμπεριλαμβάνεται στον Sniper και χρησιμοποιείται για την εξαγωγή των στατιστικών μέσα από μία προσομοίωση που έχει ολοκληρωθεί.

**SPEC_CPU2006 benchmarks**

Τα μετροπρογράμματα τα οποία χρησιμοποιήθηκαν για την εκτέλεση της άσκησης είναι ορισμένα είναι ορισμένα SPEC_CPU2006 benchmarks. Πιο συγκεκριμένα θα χρησιμοποιηθούν τα παρακάτω 12 benchmarks:

403.gcc
429.mcf
434.zeusmp
436.cactusADM
445.gobmk
450.soplex
456.hmmer
458.sjeng
459.GemsFDTD
471.omnetpp
473.astar
483.xalancbmk.

Ωστόσο, επειδή η εκτέλεση με χρήση του sniper είναι πολύ αργή, δεν εκτελέστηκαν ολόκληρα τα benchmarks αλλά χρησιμοποιήθηκαν τα pinballs που παρέχονται στη σελίδα του sniper.

**Θεωρητικό Υπόβαθρο**

Οι υπερβαθμωτοί (superscalar) επεξεργαστές επιτυγχάνουν παραλληλισμό στην εκτέλεση των εντολών χρησιμοποιώντας πολλές ανεξάρτητες διοχετεύσεις (pipelines), με αποτέλεσμα κάθε χρονική στιγμή σε κάθε διοχέτευση να μπορεί να εκτελείται και διαφορετική εντολή. Ως συνέπεια, ένας superscalar επεξεργαστής εκτελεί περισσότερες από μια εντολές κατά την διάρκεια ενός κύκλου ρολογιού, προωθώντας ταυτόχρονα πολλές εντολές σε πολλαπλά δομικά στοιχεία στον ίδιο επεξεργαστή (διαθλετει πολλαπλά functional units). Ο επεξεργαστής μπορεί να εκτελέσει παράλληλα πολλές εντολές μηχανής. Αντίθετα, εντολές που επηρεάζουν η μία την άλλη εκτελούνται σειριακά. Το μεγαλύτερο πρόβλημα της εκτέλεσης πολλαπλών εντολών μηχανής σε μια υπερβαθμωτή αρχιτεκτονική, είναι η αντιμετώπιση των εξαρτήσεων δεδομένων. (data dependencies). Τα βασικά χαρακτηριστικά ενός επεξεργαστή υπερβαθμωτής αρχιτεκτονικής είναι τα παρακάτω:

- Window size : Ο αριθμός των εντολών μηχανής που βρίσκονται σε αναμονή, από τις οποίες αυτές που είναι έτοιμες επιλέγονται για έκδοση.
- Dispatch/issue width : Ο αριθμός των εντολών μηχανής που μπορούν να αποσταλούν για εκτέλεση σε κάθε κύκλο μηχανής.

**Πειραματική Αξιολόγηση:**

Αρχικά, εκτελέστηκαν όλα τα benchmarks για κάθε διαφορετικό επεξεργαστή που προκύπτει από το συνδυασμό των παρακάτω τιμών για τις παραμέτρους dispatch_width και window_size:

| dispatch_width | 1 | 2 | 4 | 8 | 16 | 32 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| window_size | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 96 | 128 | 192 | 256 | 384 |

**ΕΡΩΤΗΣΗ: (i) Χρειάζεται πραγματικά να προσομοιώσετε και τους 60 διαφορετικούς επεξεργαστές που προκύπτουν με βάση τις παραπάνω τιμές; Αν όχι, εξηγήστε ποιές περιπτώσεις μπορείτε να παραλείψετε και γιατί. Δικαιολογήστε την απάντησή σας όχι μόνο θεωρητικά αλλά και αποδεικνύοντας για ένα μικρό αριθμό αυτών των περιπτώσεων ότι καλώς τις παραλείψατε.**

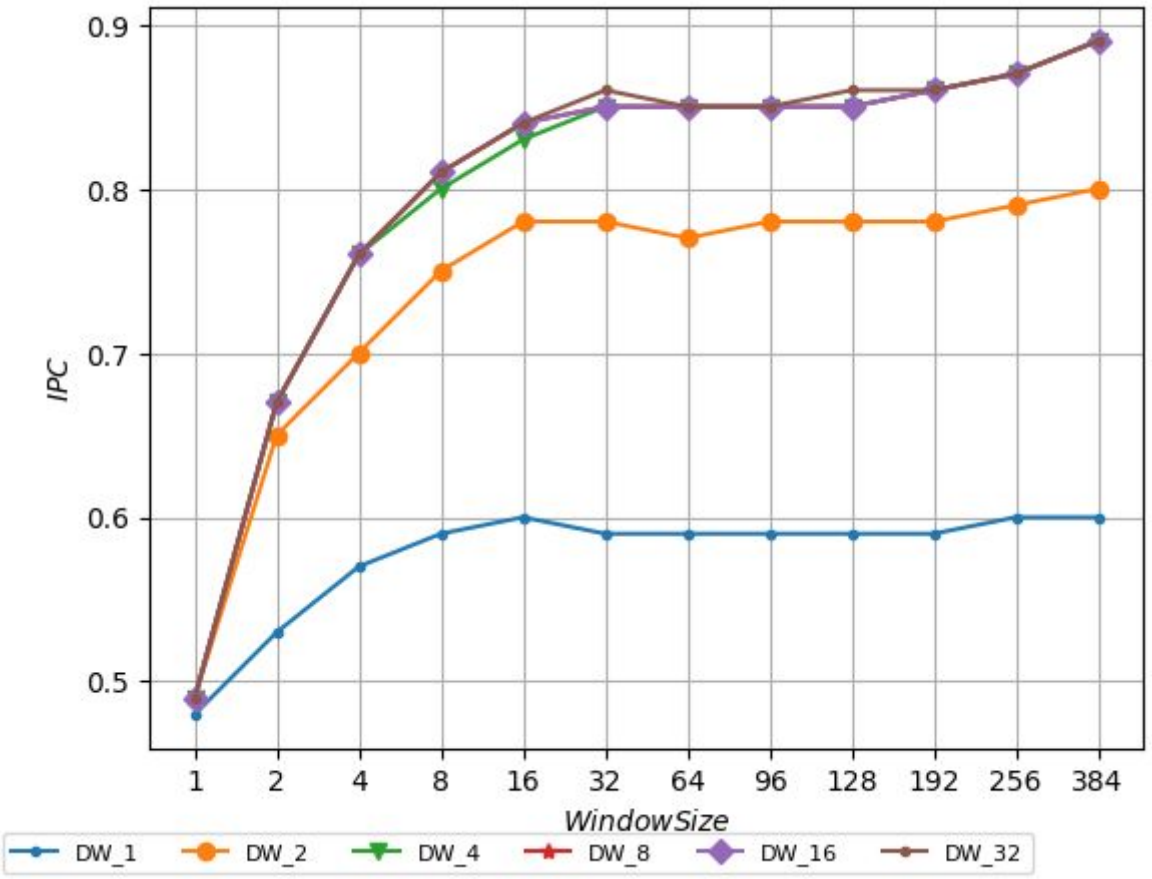
Οι συνολικοί συνδυασμοί επεξεργαστών που προκύπτουν είναι 72. Όμως, δεν είναι πραγματικά απαραίτητο να τους προσομοιώσουμε όλους, καθώς δεν έχει νόημα να προσομοιώσουμε τους επεξεργαστές όπου, **window_size < dispatch_width**. Επομένως, θα μπορούσαμε να παραλείψουμε τις προσωμειώσεις για αυτές τις περιπτώσεις και να τρέξουμε τις υπόλοιπες 57 περιπτώσεις αντί για τις 72 συνολικές περιπτώσεις.
Πράγματι, τρέχοντας το πρώτο benchmark και για τους 72 διαφορετικούς επεξεργαστές, επιβεβαιώνεται η απόφασή μας αυτή.
Σημείωση: Σημειώνεται πως για την πληρότητα της άσκησης σημειώθηκαν σε όλα τα benchmarks και οι 72 προσωμειώσεις.

**Αξιολόγηση της επίδοσης**

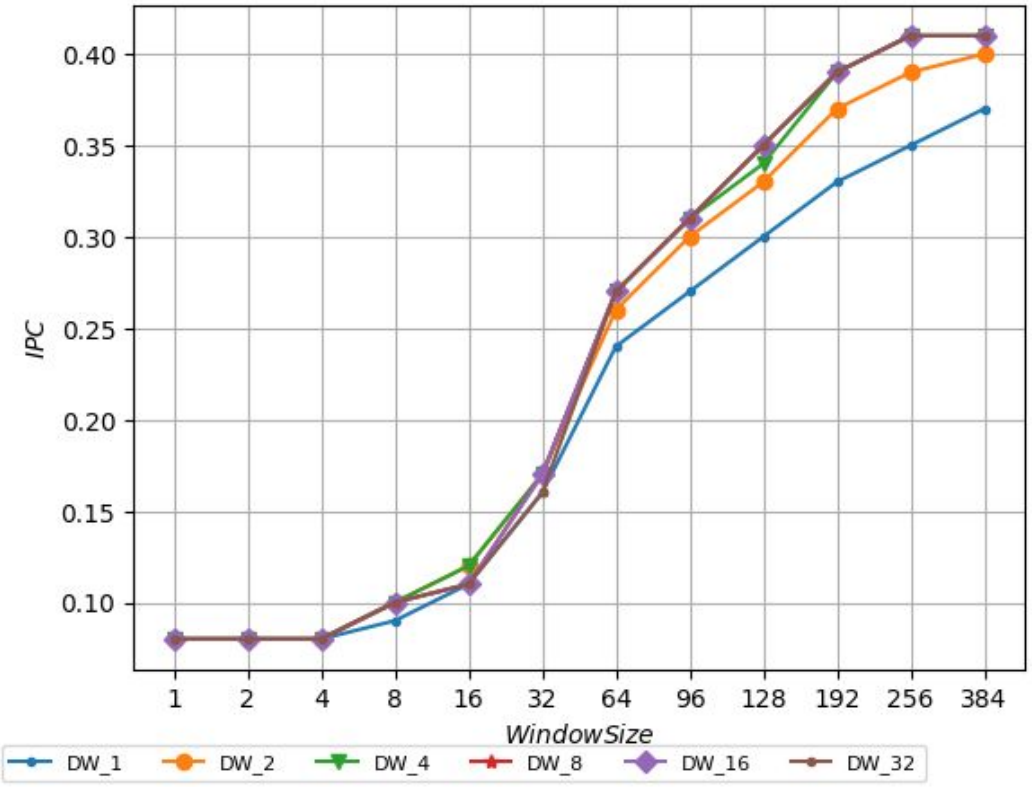
Αρχικά θα παρουσιάσουμε το αποτέλεσμα και των 72 προσομοιώσεων έτσι ώστε να δείξουμε ότι δεν προσφέρουν κάποια παραπάνω πληροφορία σε σχέση με τις υπόλοιπες 57 καταγραφές στις οποίες ισχύει ότι **window_size >= dispatch_width.**

Η μετρική ως προς την οποία αξιολογήθηκε η απόδοση είναι το IPC (Instructions per Cycle) και η παρουσίαση θα γίνει ανά benchmark:

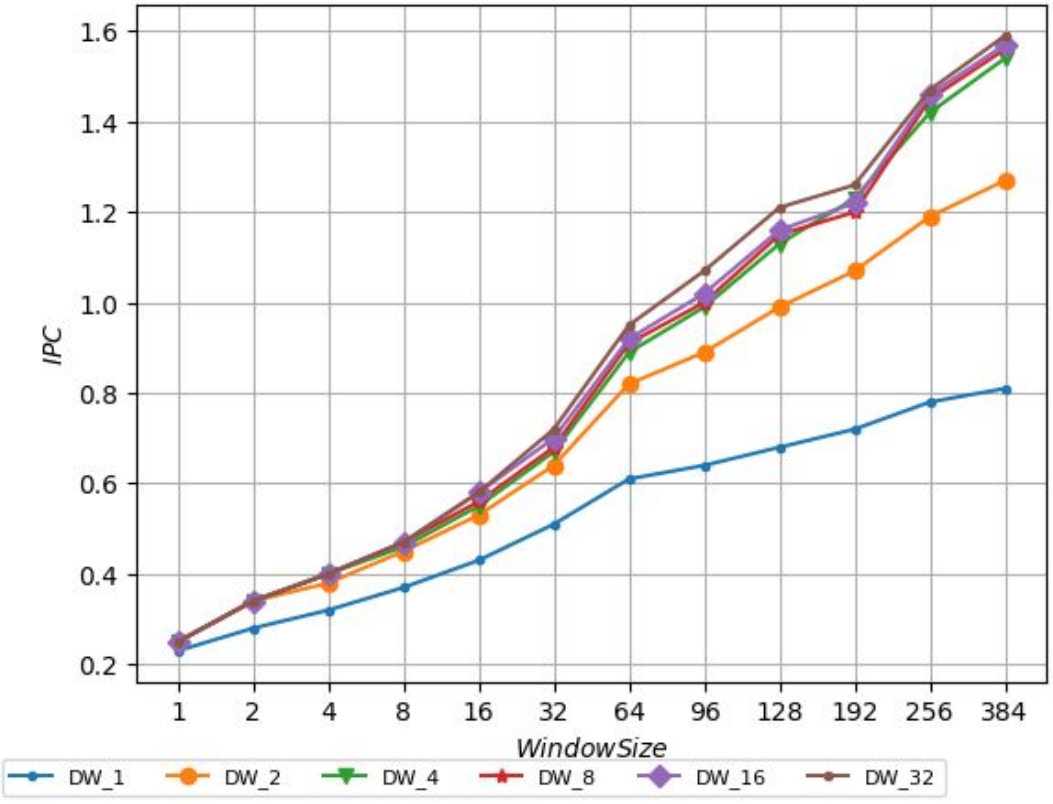
**gcc**



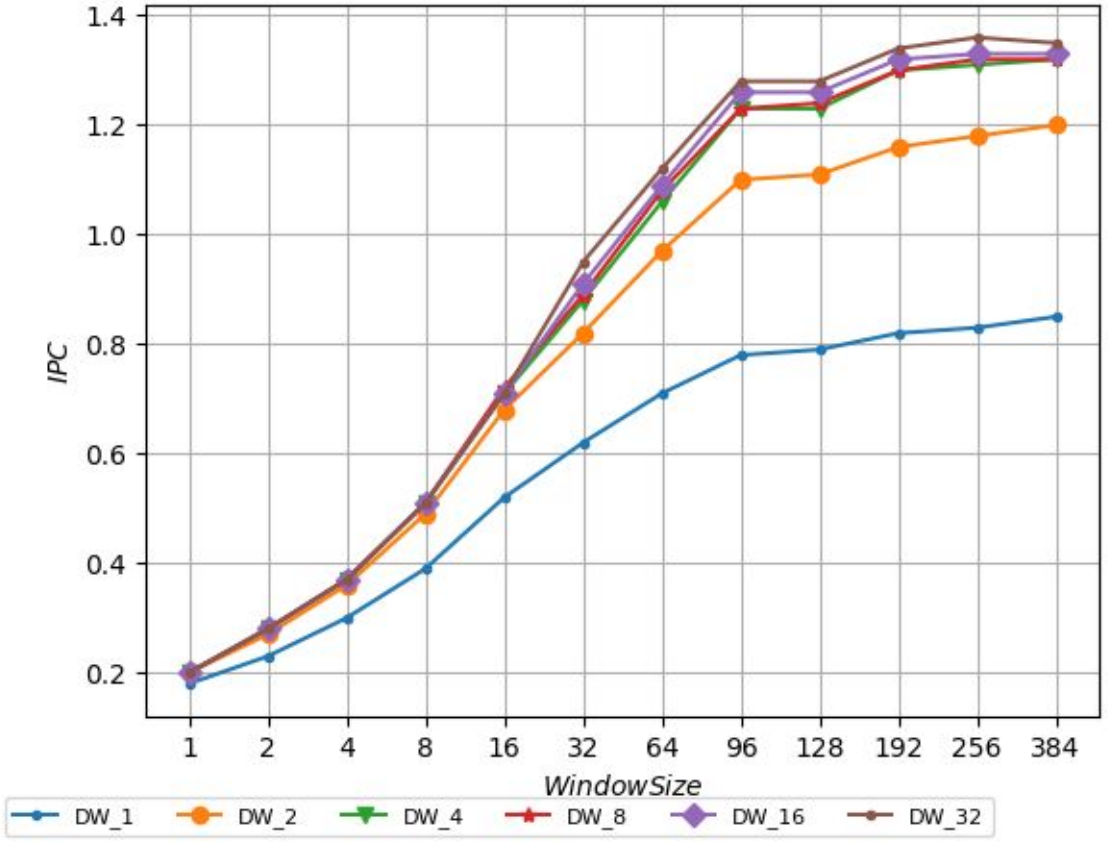
**mcf**


0.40
0.35
0.30
0.25
0.20
0.15
0.10
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

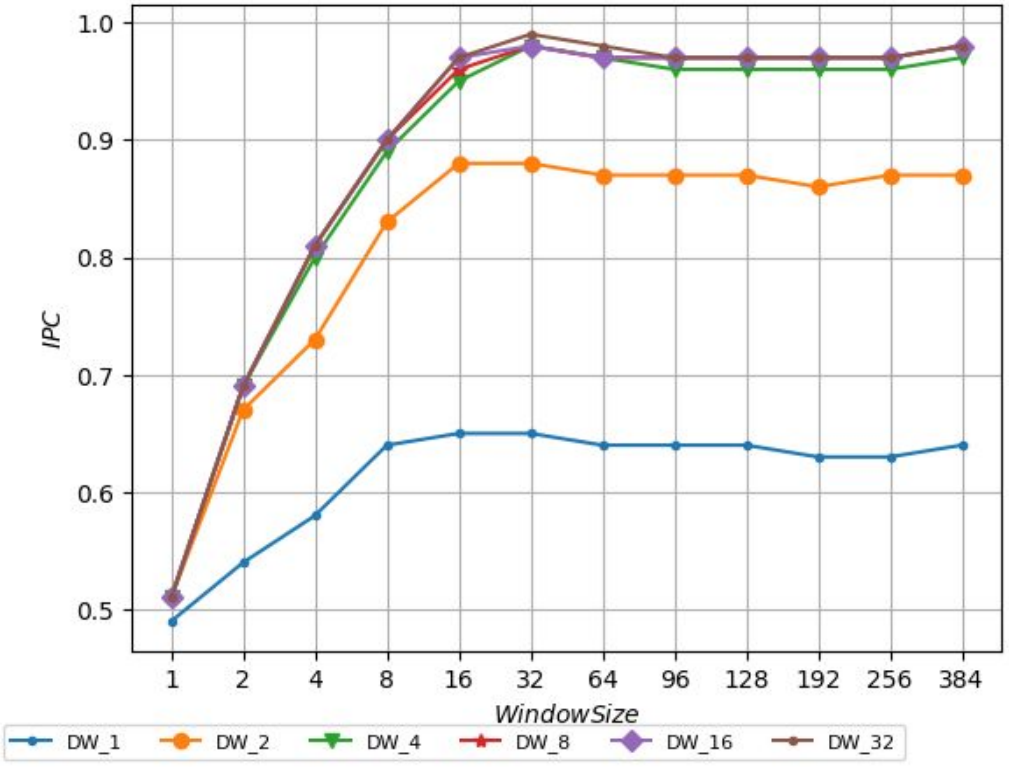
**zeusmp**


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

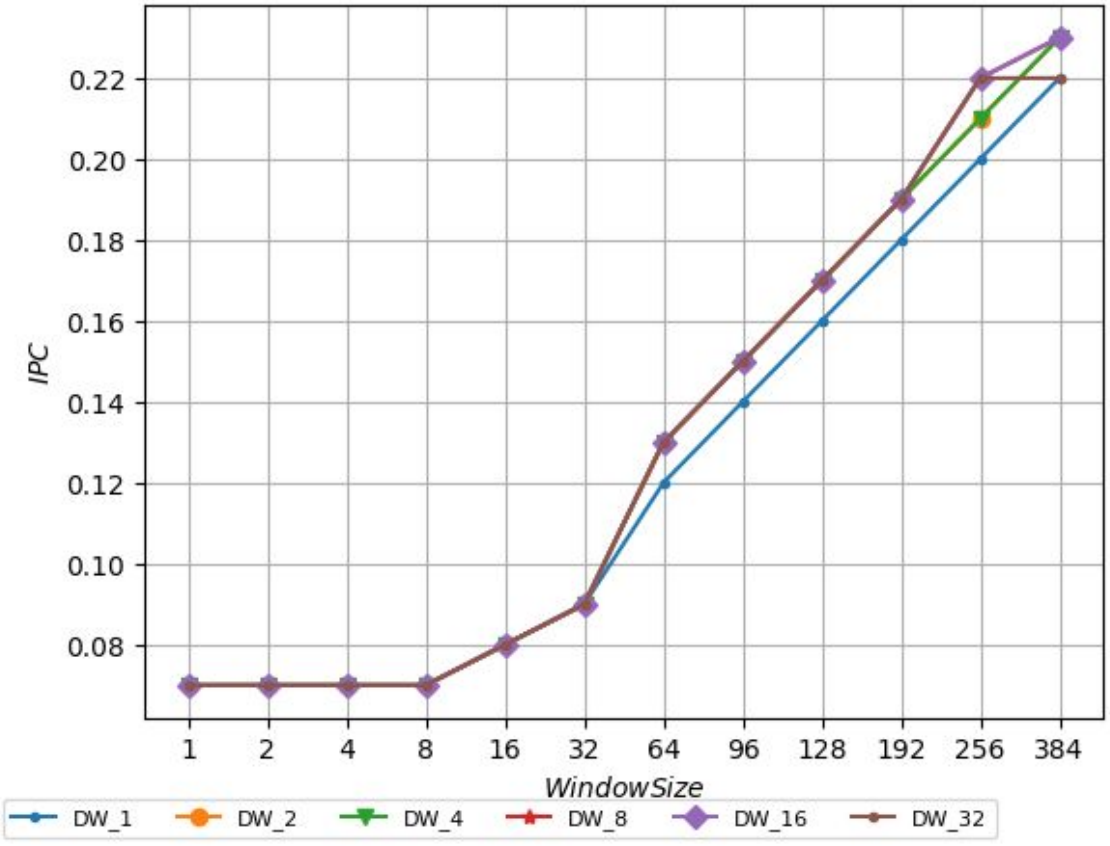
**cactusADM**


1.4
1.2
1.0
0.8
0.6
0.4
0.2
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

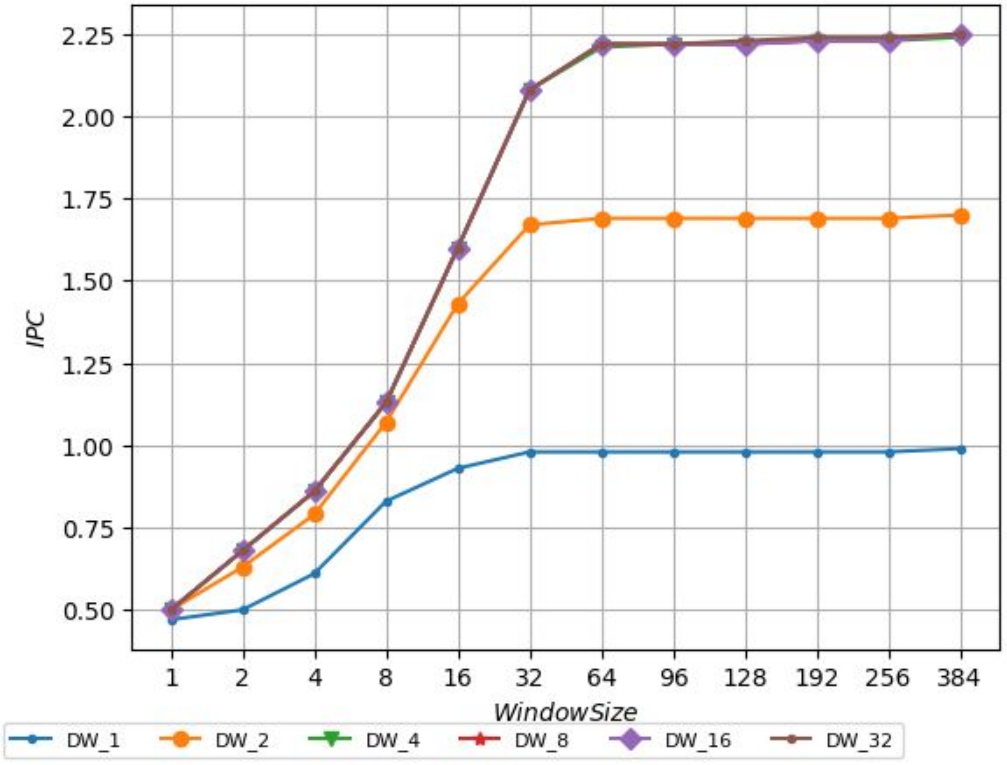
**gobmk**


1.0
0.9
0.8
0.7
0.6
0.5
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

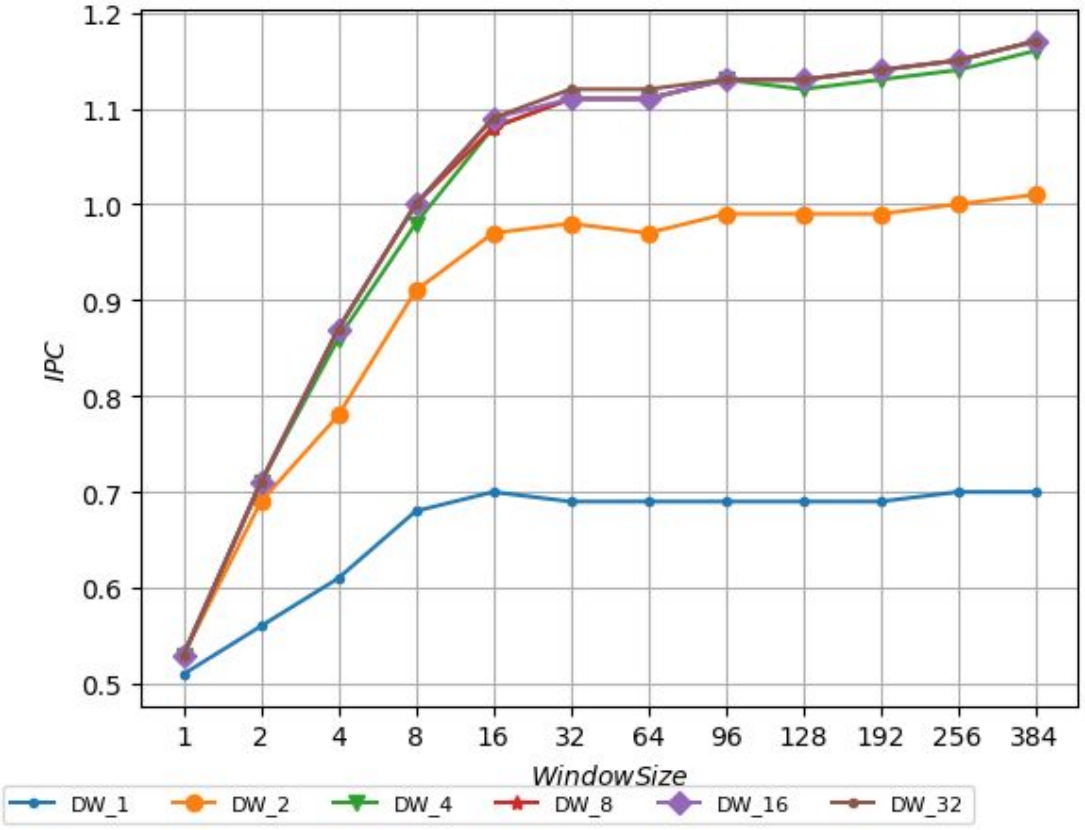
**soplex**


0.22
0.20
0.18
0.16
0.14
0.12
0.10
0.08
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

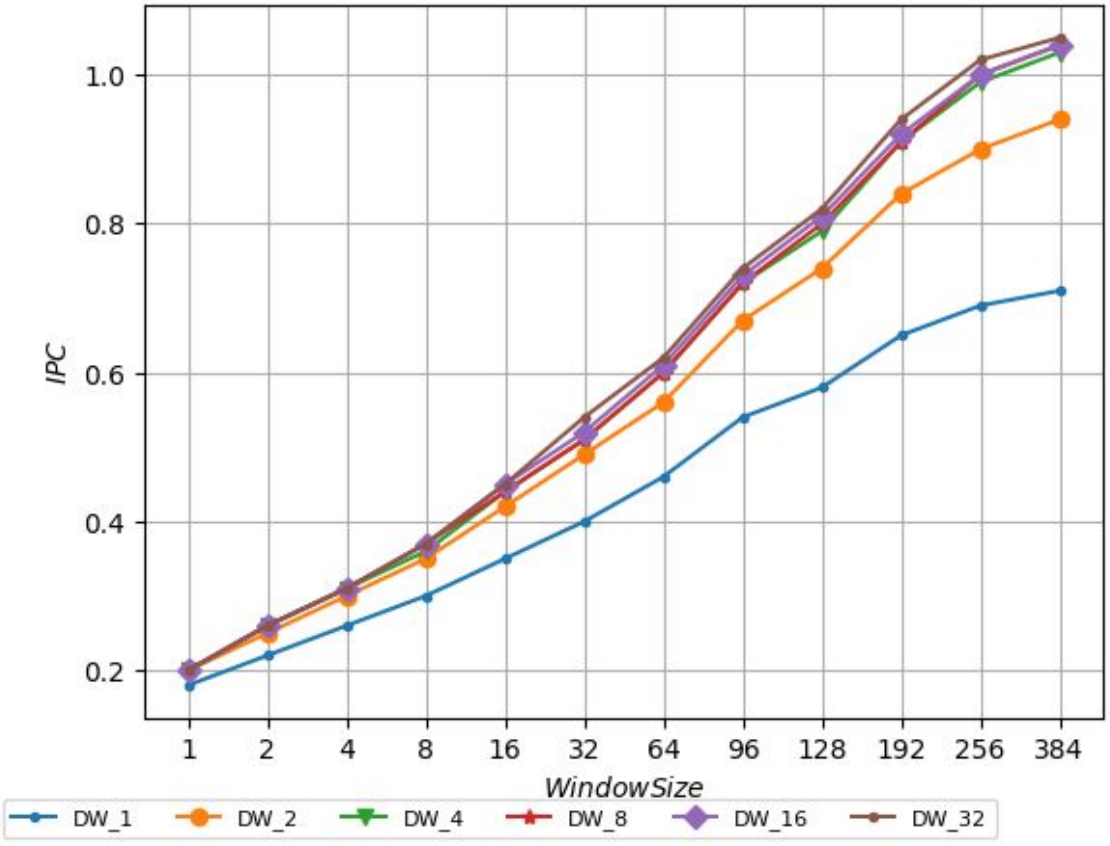
**hmmer**


2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

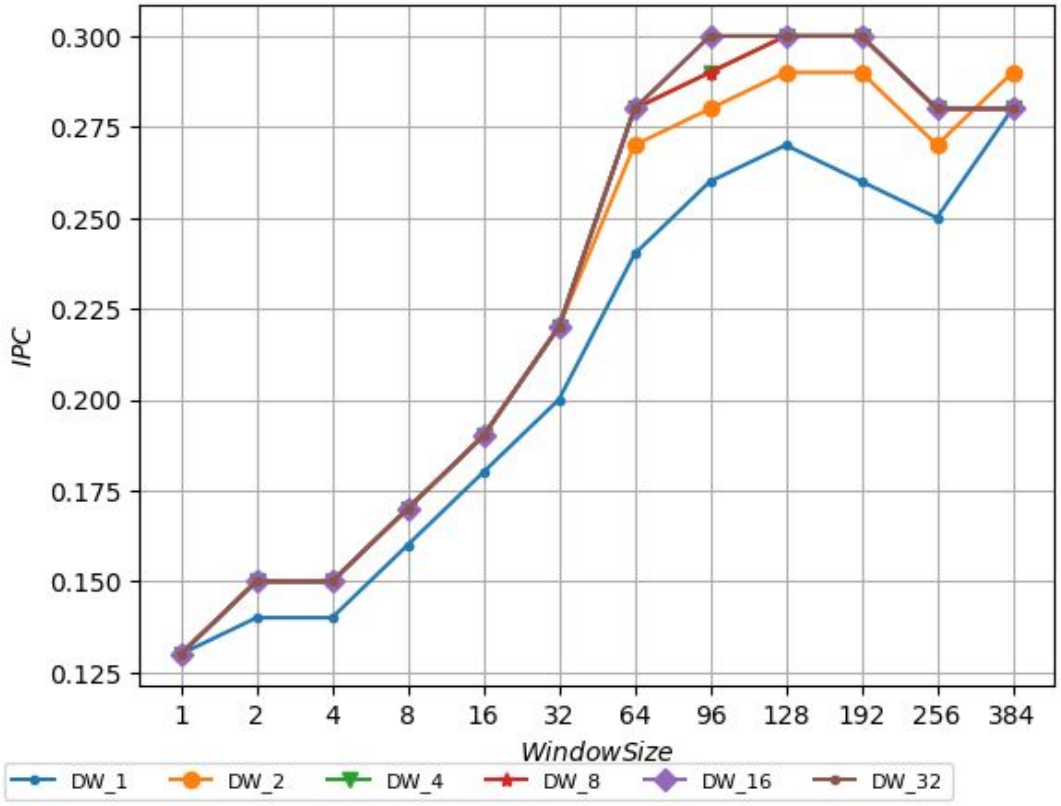
**sjeng**


1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

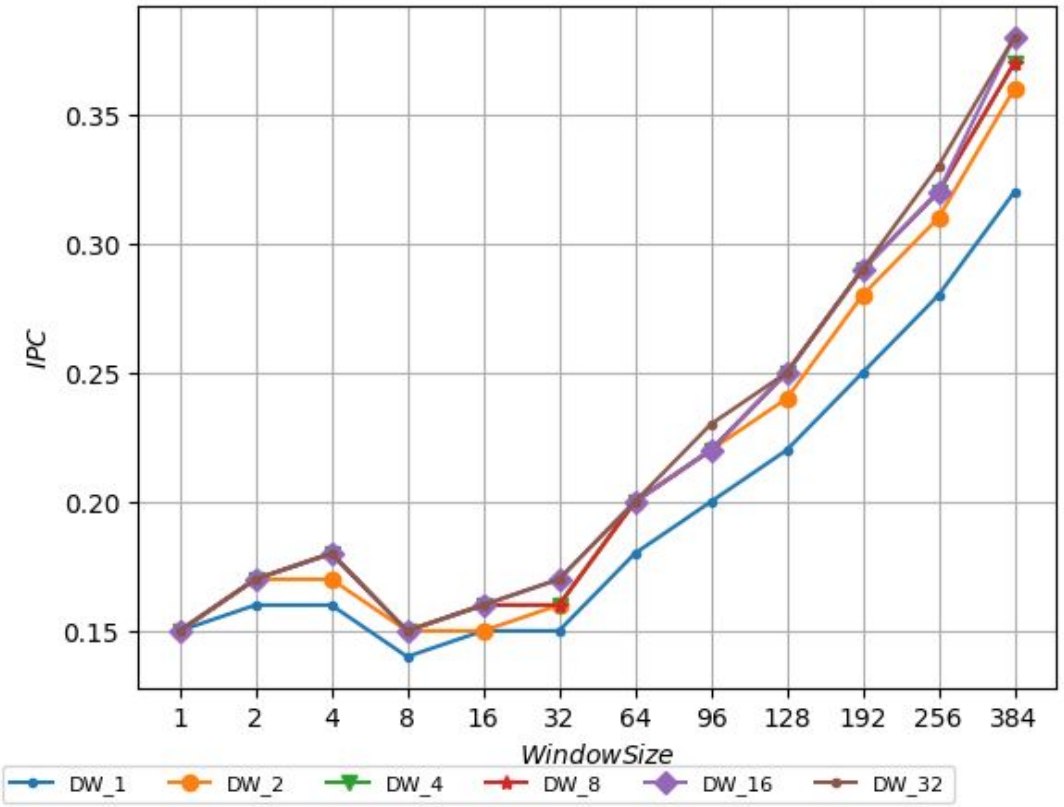
**GemsFDTD**


1.0
0.8
0.6
0.4
0.2
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

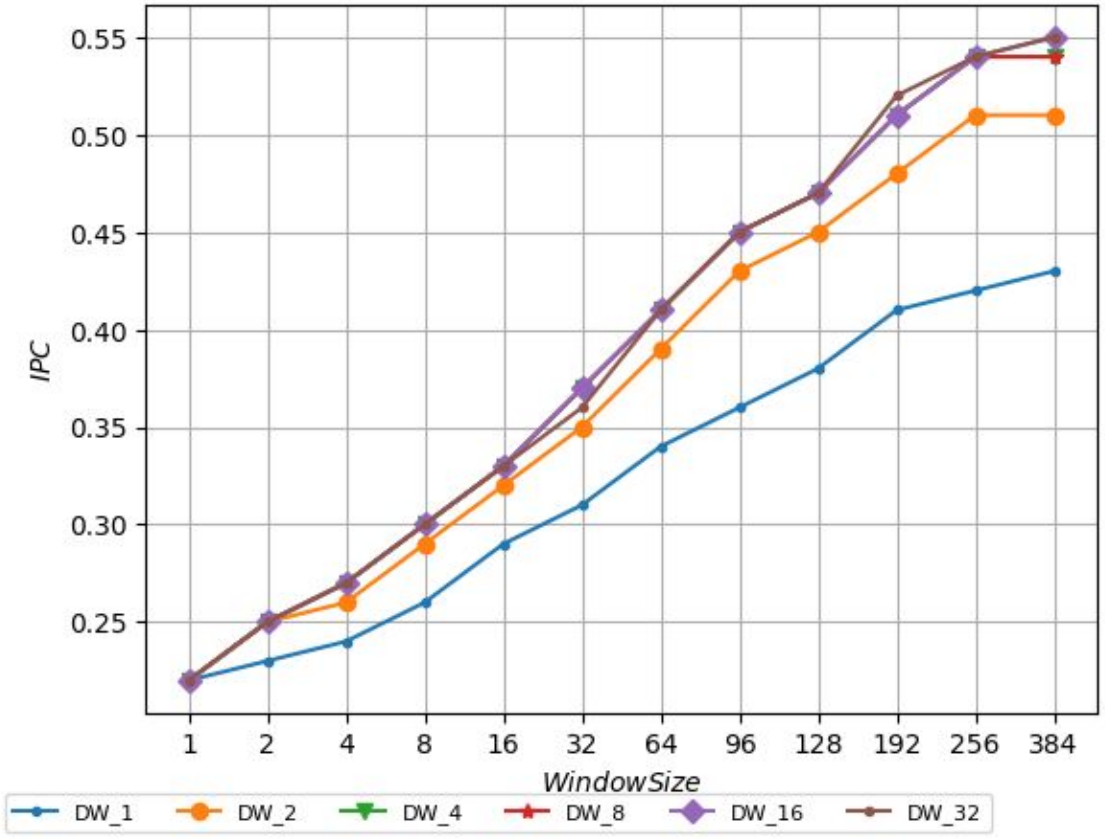
**omnetpp**


0.300
0.275
0.250
0.225
0.200
0.175
0.150
0.125
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**astar**

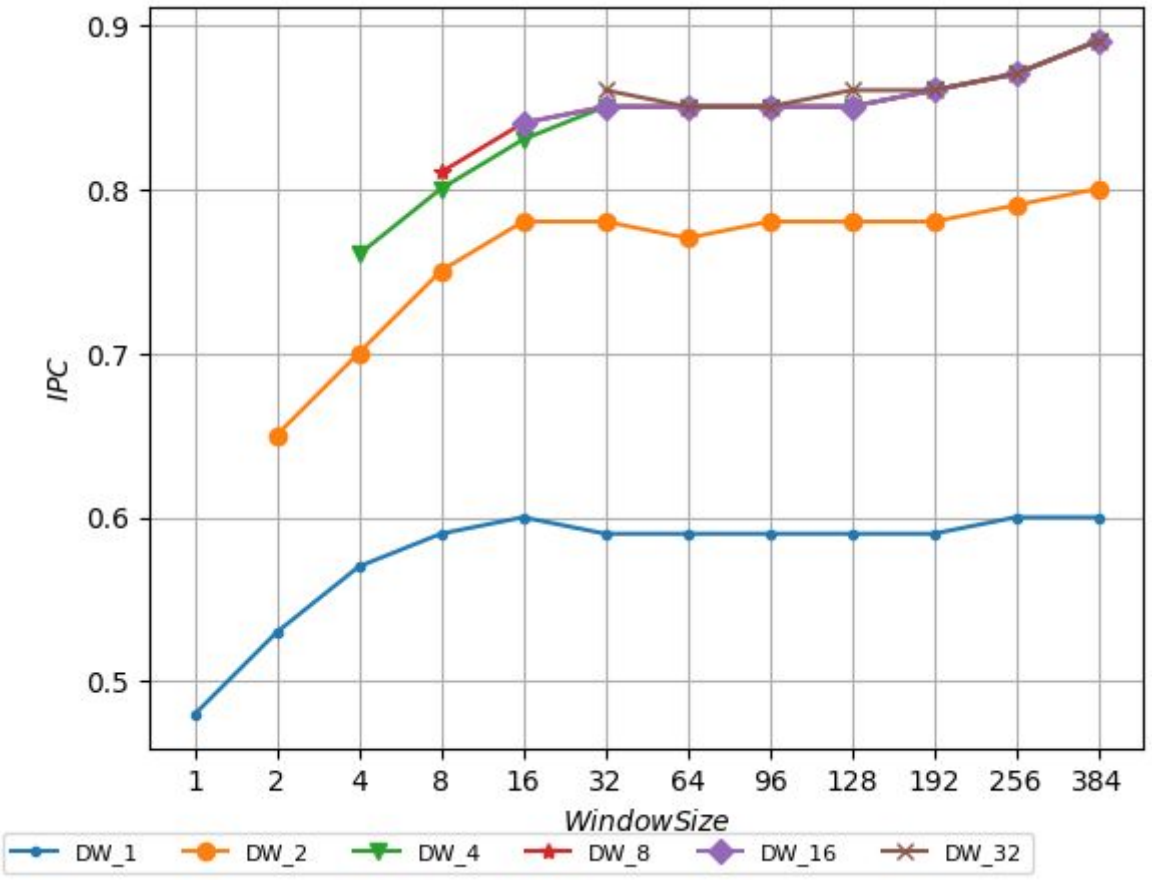

0.35
0.30
0.25
0.20
0.15
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**xalancbmk**

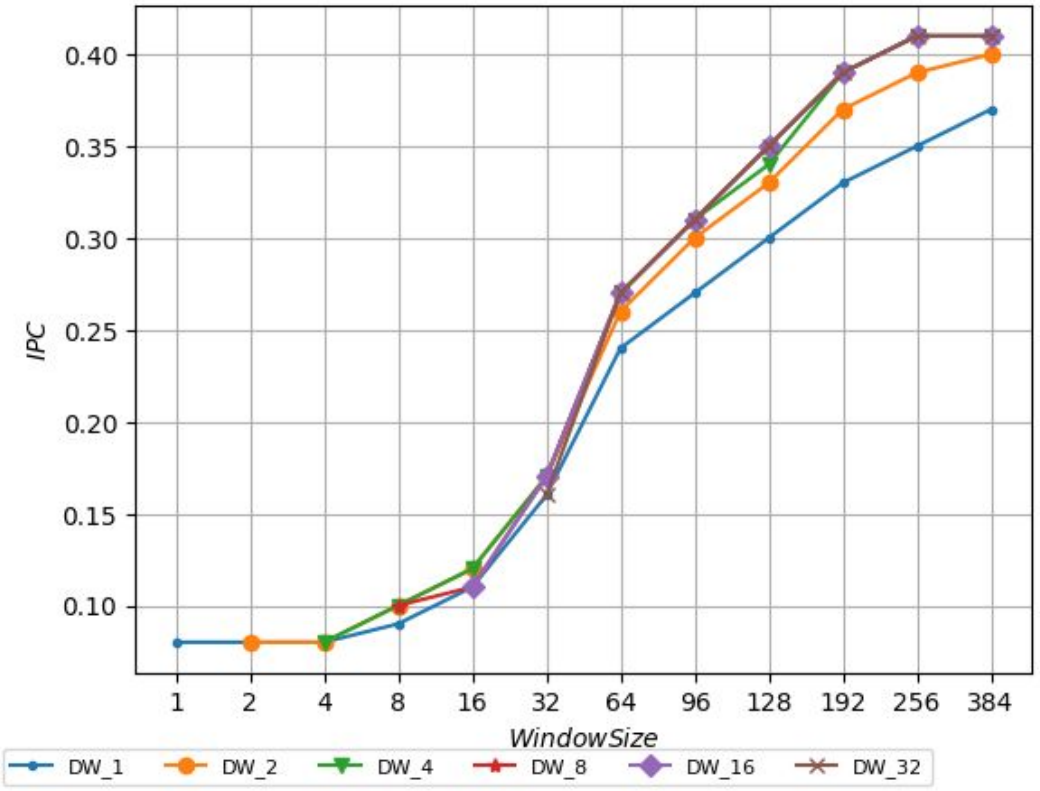

0.55
0.50
0.45
0.40
0.35
0.30
0.25
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

Στην συνέχεια παρουσιάζουμε τα αποτελέσματα μόνο των 57 προσομοιώσεων για τις οποίες ισχύει ότι window_size >=  dispatch_width, όπου παρατηρείται αυτό που αναμέναμε από την θεωρία.

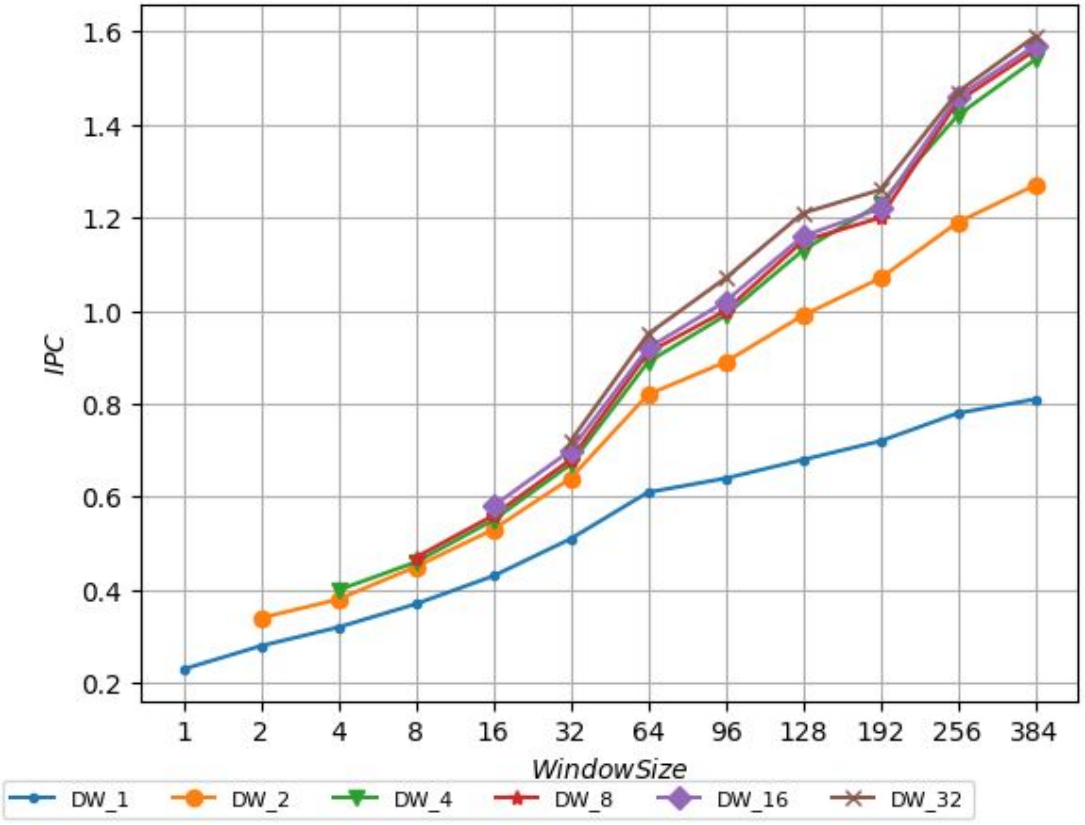
**gcc**



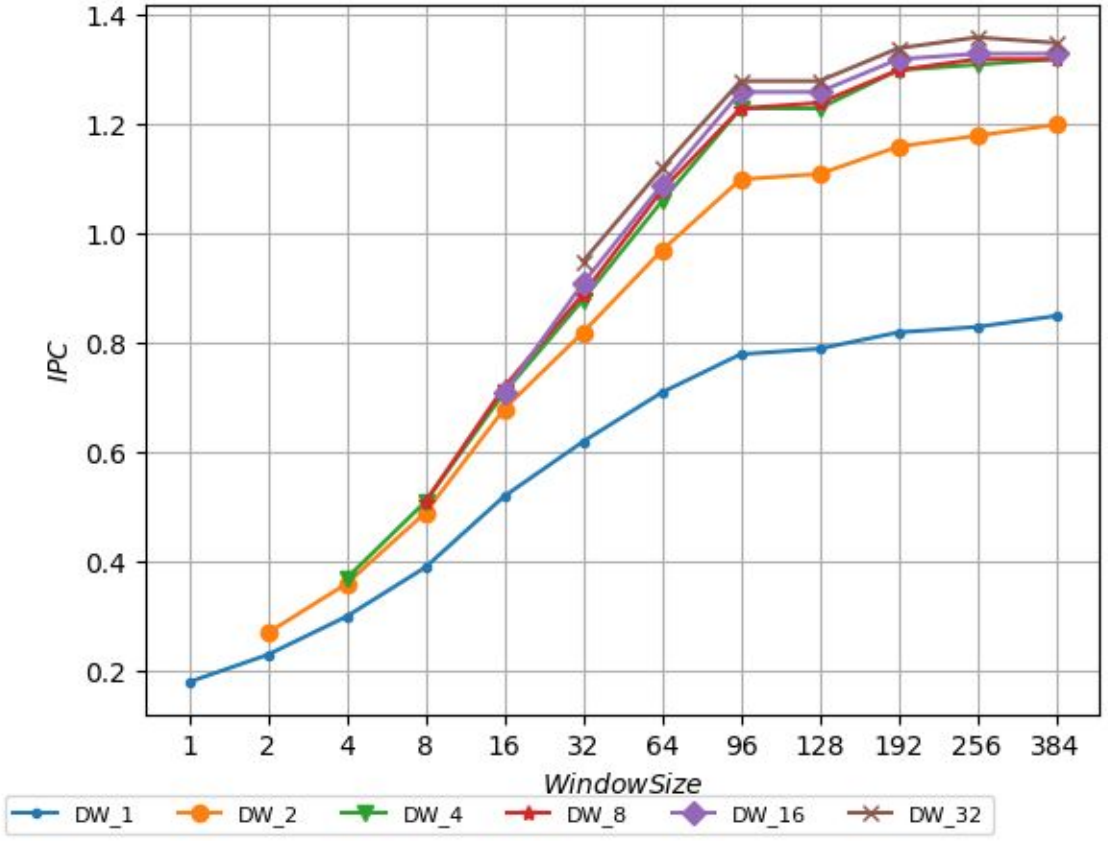
**mcf**


0.40
0.35
0.30
0.25
0.20
0.15
0.10
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

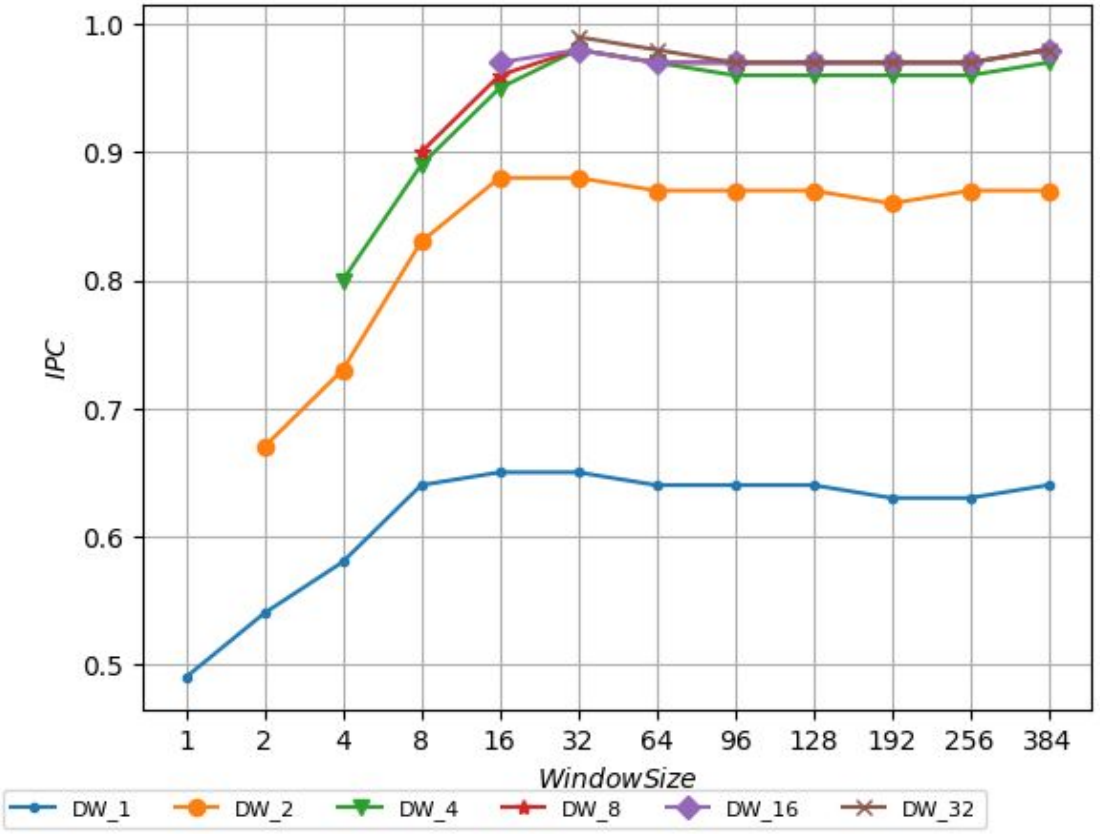
**zeusmp**


1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

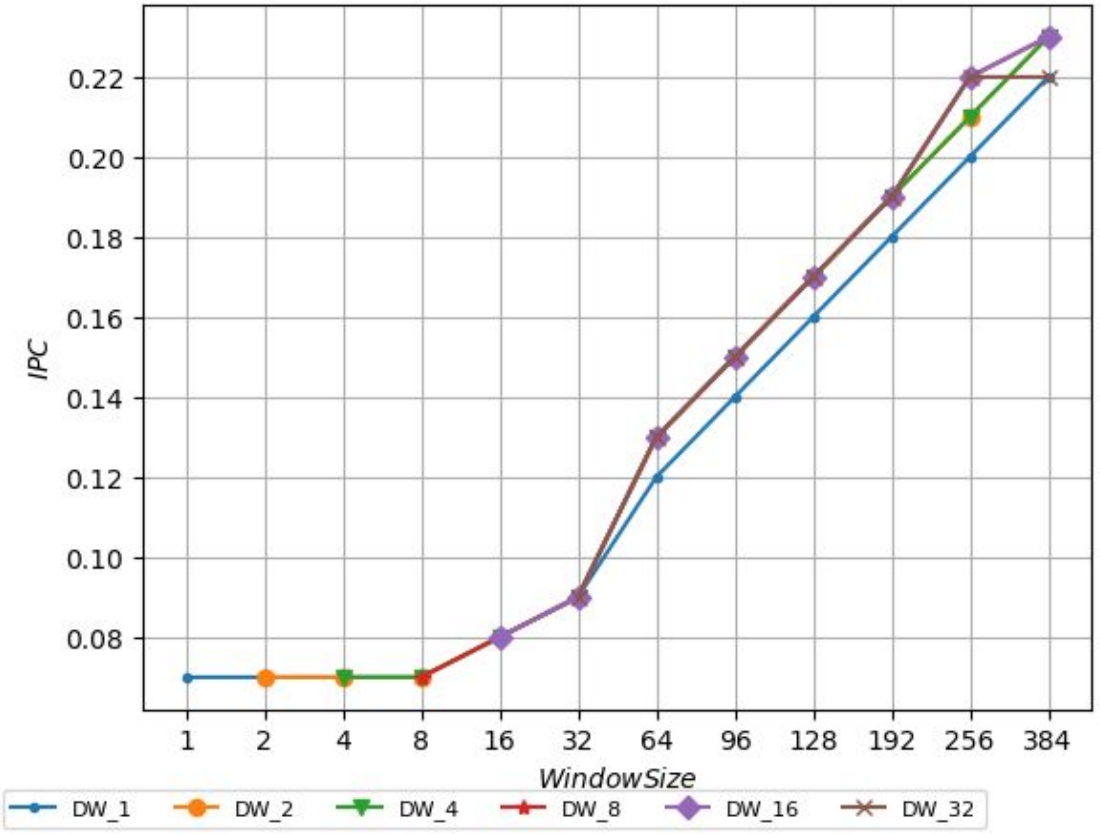
**cactusADM**


1.4
1.2
1.0
0.8
0.6
0.4
0.2
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

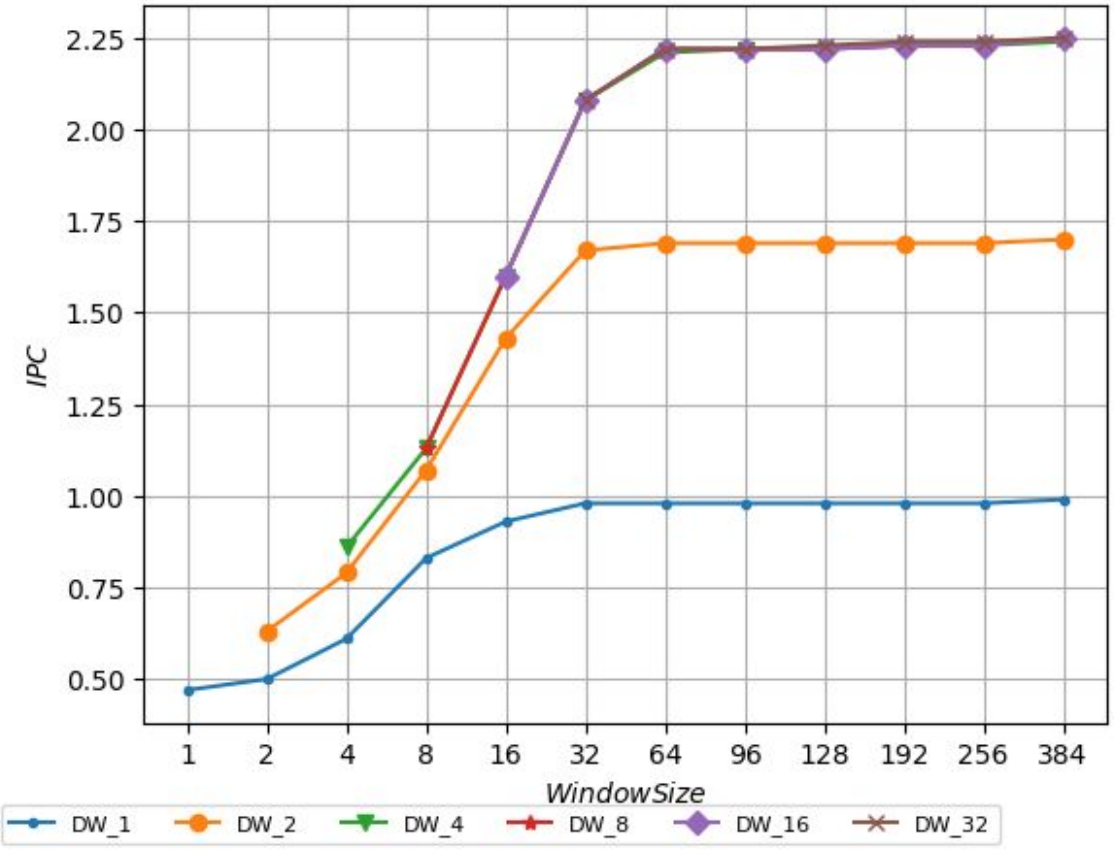
**gobmk**


1.0
0.9
0.8
0.7
0.6
0.5
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

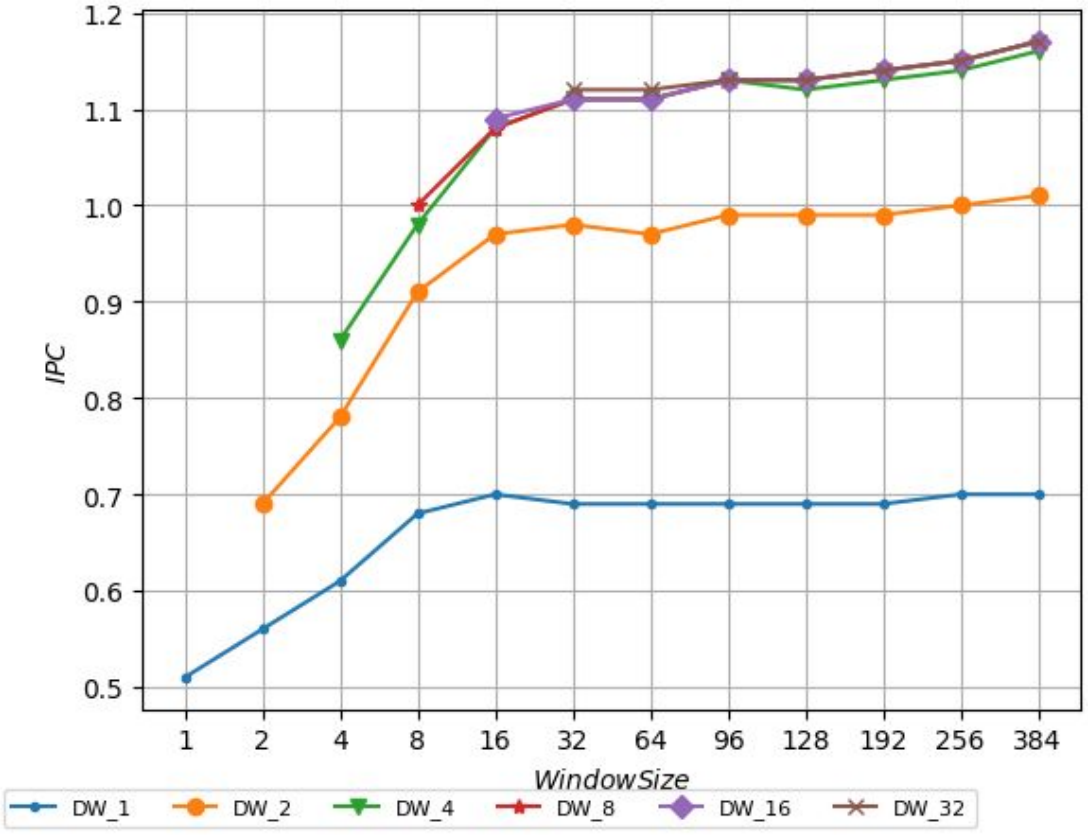
**soplex**


0.22
0.20
0.18
0.16
0.14
0.12
0.10
0.08
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

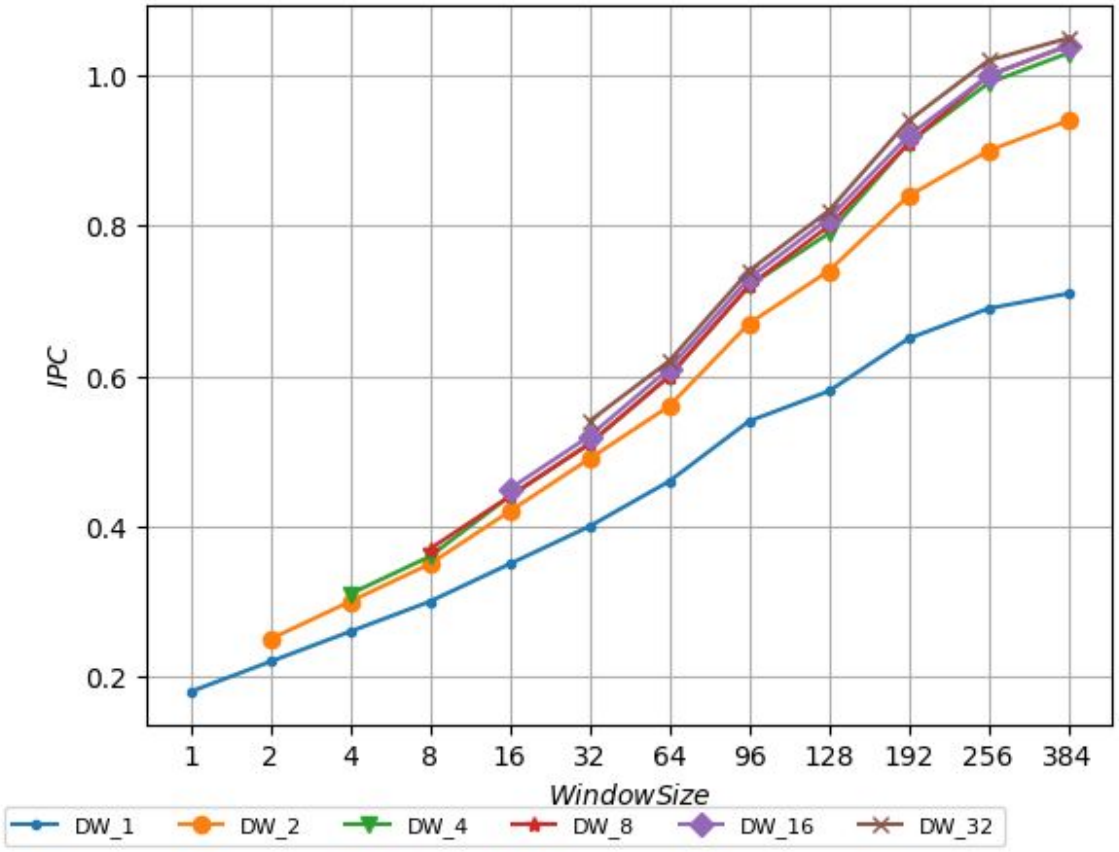
**hmmer**


2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

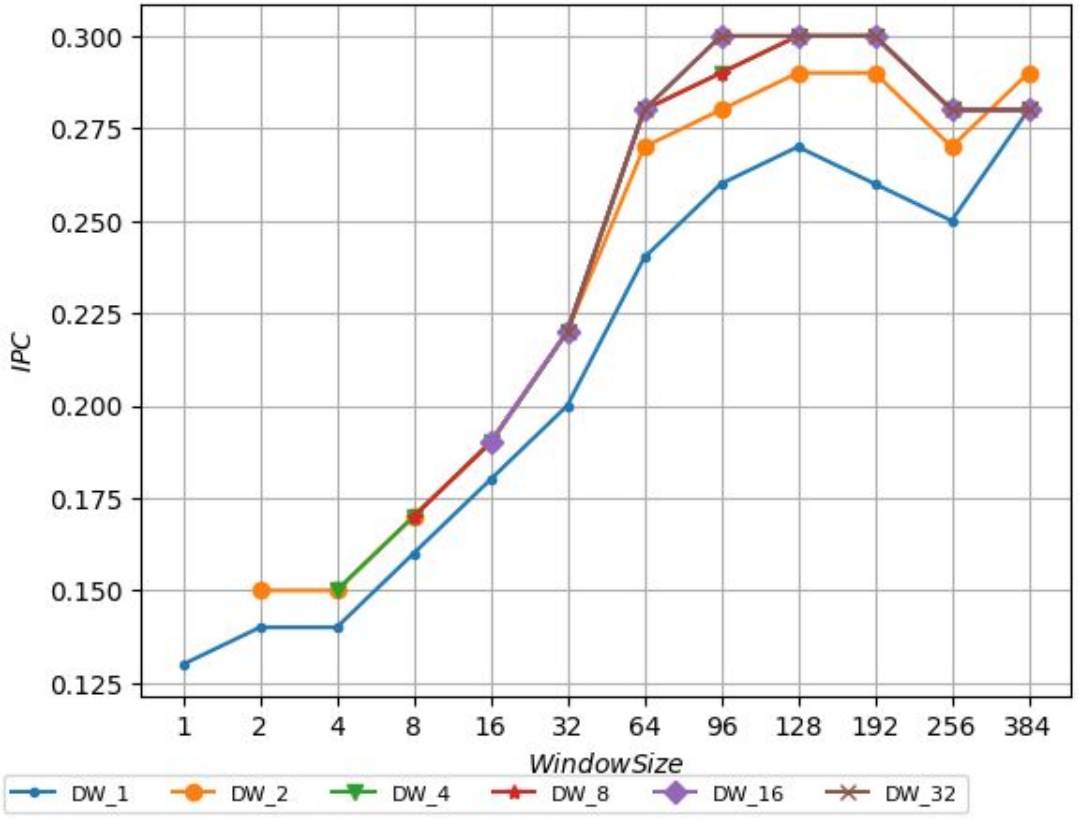
**sjeng**


1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

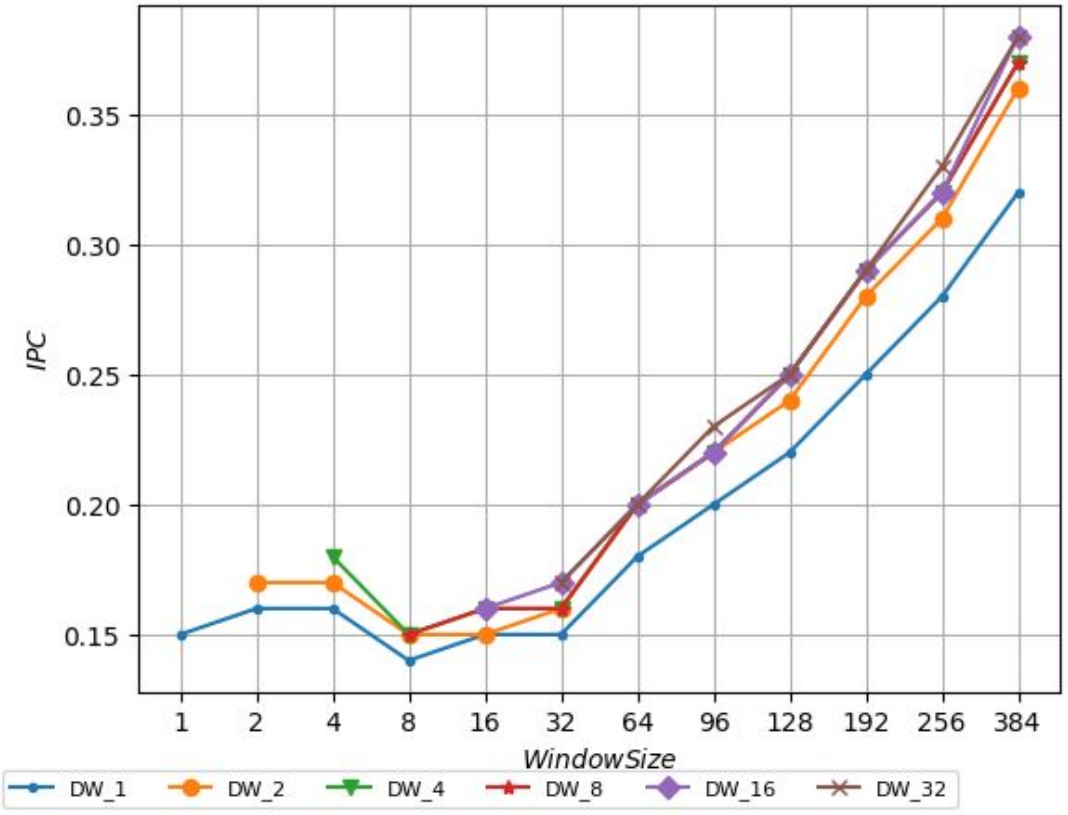
**GemsFDTD**


1.0
0.8
0.6
0.4
0.2
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

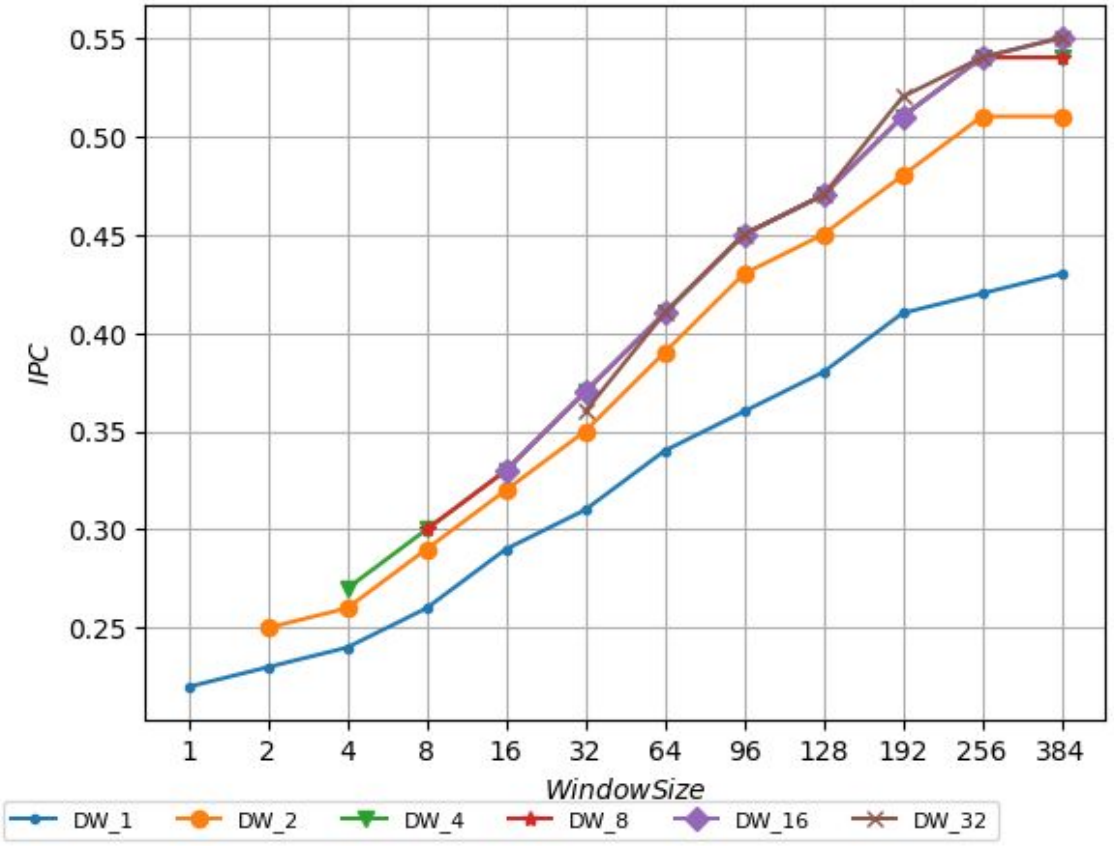
**omnetpp**


0.300
0.275
0.250
0.225
0.200
0.175
0.150
0.125
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**astar**


0.35
0.30
0.25
0.20
0.15
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**xalancbmk**


0.55
0.50
0.45
0.40
0.35
0.30
0.25
IPC
1
2
4
8
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**ΕΡΩΤΗΣΗ (ii) Πώς επηρεάζει η κάθε παράμετρος την απόδοση του επεξεργαστή; Σε ποιά συμπεράσματα μπορείτε να καταλήξετε ως προς το σχεδιασμό ενός superscalar, out-of-order επεξεργαστή;**

Από τα παραπάνω αποτελέσματα των προσωμειώσεων που διεξήχθησαν καταλήγουμε στα παρακάτω συμπεράσματα.

Αρχικά εν γένει, η αύξηση του window size συνεπάγεται αύξηση της απόδοσης.
Μπορούμε, εδώ, να διακρίνουμε τεσσερεις διαφορετικές συμπεριφορές στα benchmarks που εξετάσαμε.

- Η συνηθέστερη συμπεριφορά είναι η αύξηση του window size να συνεπάγεται αύξηση του IPC όσο και αν το αυξήσουμε.
  Αυτο το παρατηρούμε για τα εξείς benchmarks: zeusmp, soplex, GemsFDTD, astar.
- Σε ορισμένα benchmarks η αύξηση του IPC σταματάει για Window_Size ίσο με 16 και στη συνέχεια σταθεροποιείται.
  Αυτο το παρατηρούμε για τα εξείς benchmarks: gcc, gobmk,hmmer,sjeng
- Σε ορισμένα benchmarks η αύξηση του IPC σταματάει για Window_Size ίσο με 96 και στη συνέχεια σταθεροποιείται.
  Αυτο το παρατηρούμε για τα εξείς benchmarks: cactusADM, omnetpp, ας σημειωθεί οτι για το omnetpp παρατηρέιται η σταθεροποιήση για Window_Size ίσο με 96 αλλά όταν συνεχίσει να αυξάνει για Window_Size ίσο με 192 αρχίζει και πέφτει το ipc.
- Σε ορισμένα benchmarks η αύξηση του IPC σταματάει για Window_Size ίσο με 256 και στη συνέχεια σταθεροποιείται.
  Αυτο το παρατηρούμε για τα εξείς benchmarks: mcf,xalancbmk

Η βελτίωση του IPC μέσω της αύξησης του window size, φαίνεται να επηρεάζεται και από το dispatch width. Όσον αφορά το Dispatch_Width, παρατηρώντας τα αποτελέσματα καταλήγουμε ότι ενώ η αύξησή του από 1 σε 2 έχει ως αποτέλεσμα βελτίωση σε πολύ μεγάλο βαθμό την απόδοση, για μεγαλύτερες τιμές υπάρχει ελάχιστη εώς σχεδόν μηδαμινή βελτίωση της απόδοσης. Αυτό πιθανότατα συμβαίνει, διότι τα benchmarks δεν είναι φτιαγμένα έτσι ώστε να εκμεταλλεύονται το μεγάλο dispatch width, επομένως αν και θεωρητικά θα έπρεπε να είχαμε βελτίωση της απόδοσης, το μεγάλο dispatch width μένει ανεκμετάλλευτο.

Κατανάλωση Ενέργειας – Μέγεθος Chip

Έχοντας εξάγει τα συμπεράσματά μας για την επίδραση των 2 παραμέτρων (Window_Size, Dispatch_Width) στην επίδοση του επεξεργαστή, παρακάτω εξετάζεται η επίδραση των παράμετρων στο μέγεθος του τσιπ και τη κατανάλωση ενέργειας του επεξεργαστή. Για να το πετύχουμε αυτό, θα χρησιμοποιήσουμε το Energy-Delay Product (EDP) που συνδυάζει το χρόνο εκτέλεσης ενός benchmark με την ενέργεια που δαπανήθηκε. Για την αξιολόγηση της κατανάλωσης ενός επεξεργαστή χρησιμοποιείται ως μετρική η συνολική κατανάλωση ενέργειας σε Joules. Παρ' όλα αυτα, πολλές φορές απαιτείται η μελέτη της επίδρασης διαφόρων χαρακτηριστικών του επεξεργαστή όχι μόνο στην κατανάλωση αλλά ταυτόχρονα και στην επίδοσή του. Επομένως θα χρησιμοπιηθεί η μετρική energy-delay product (EDP), το οποίο για την εκτέλεση ενός benchmark ορίζεται ως το γινόμενο της ενέργειας επί τον χρόνο εκτέλεσης του benchmark:

EDP = Energy(J) ∗ runtime(sec)

Επίσης, προκειμένου να δώσουμε περισσότερο βάρος στον χρόνο εκτέλεσης υψώνουμε το runtime στο τετράγωνο και στον κύβο.

Έτσι προκύπτουν τα:

ED^2P = Energy(J) ∗ runtime^2 (sec)

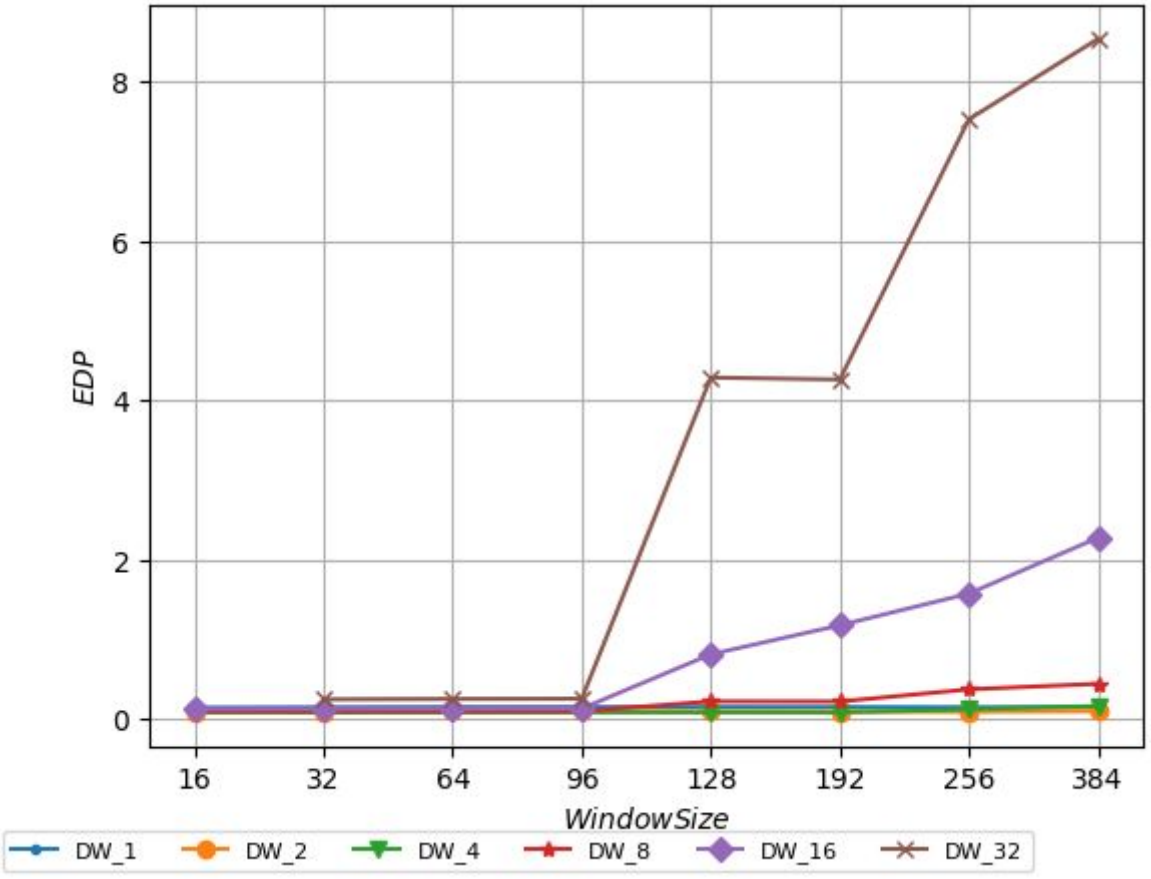
ED^3P = Energy(J) ∗ runtime^3 (sec)

Για το μέγεθος του τσιπ και τη συνολική κατανάλωση ενέργειας χρησιμοποιήθηκε το McPAT που περιλαμβάνεται στον sniper.

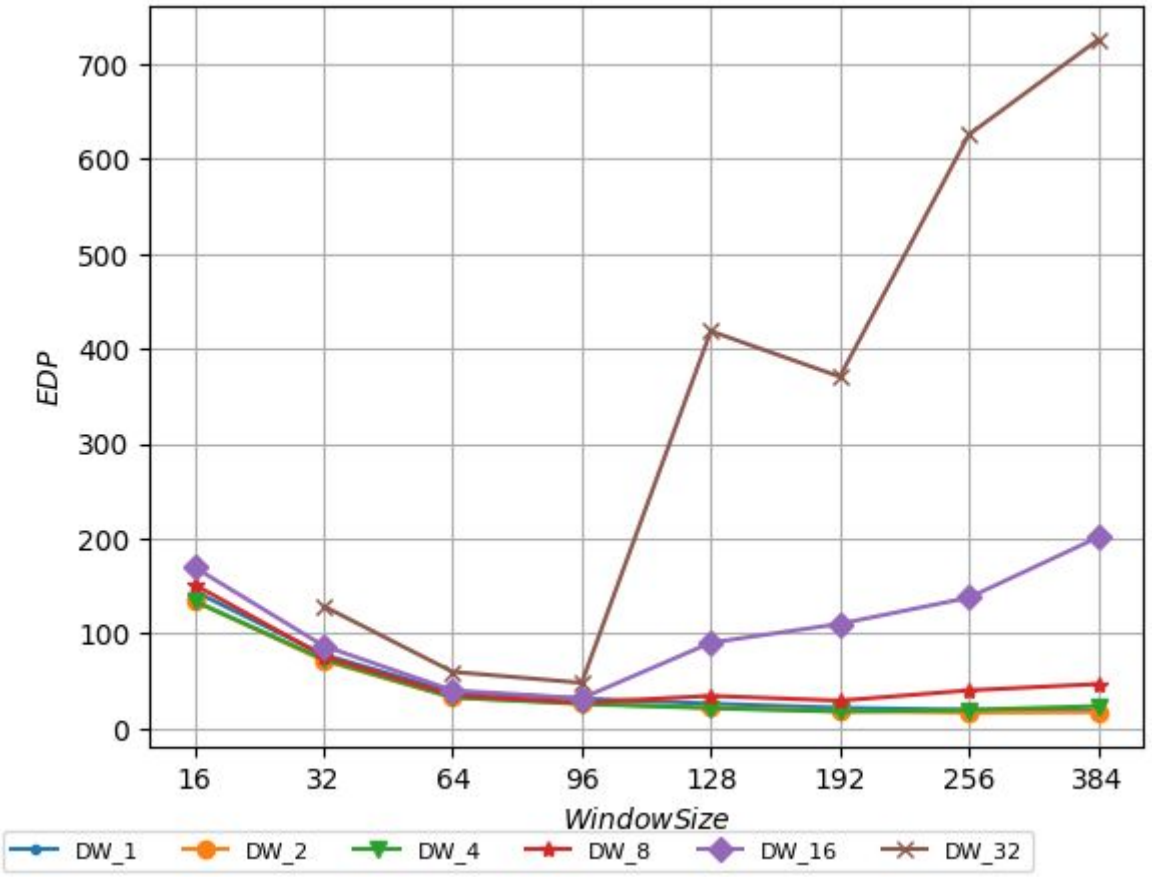
Παρακάτω παρουσιάζονται τα αποτελέσματα σε γραφικές παραστάσεις:

**Για το EDP:**

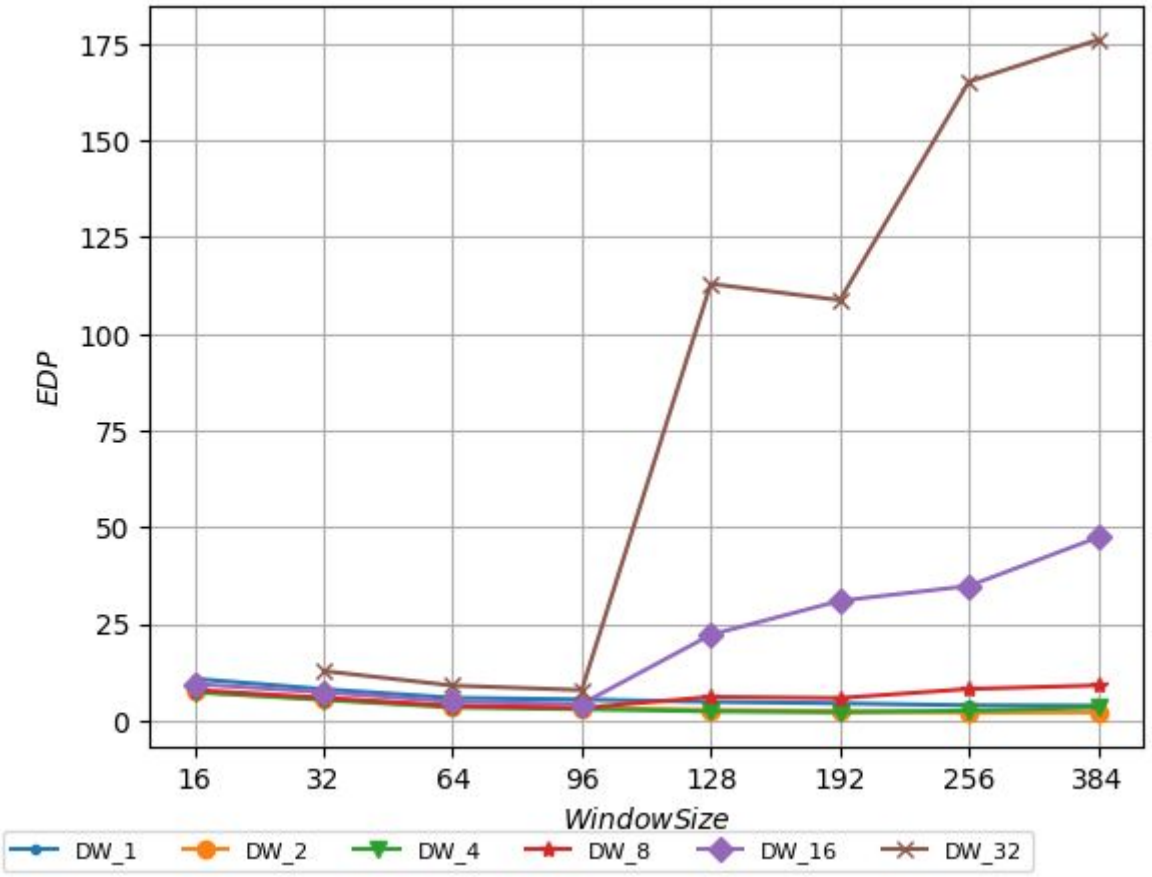
**gcc**



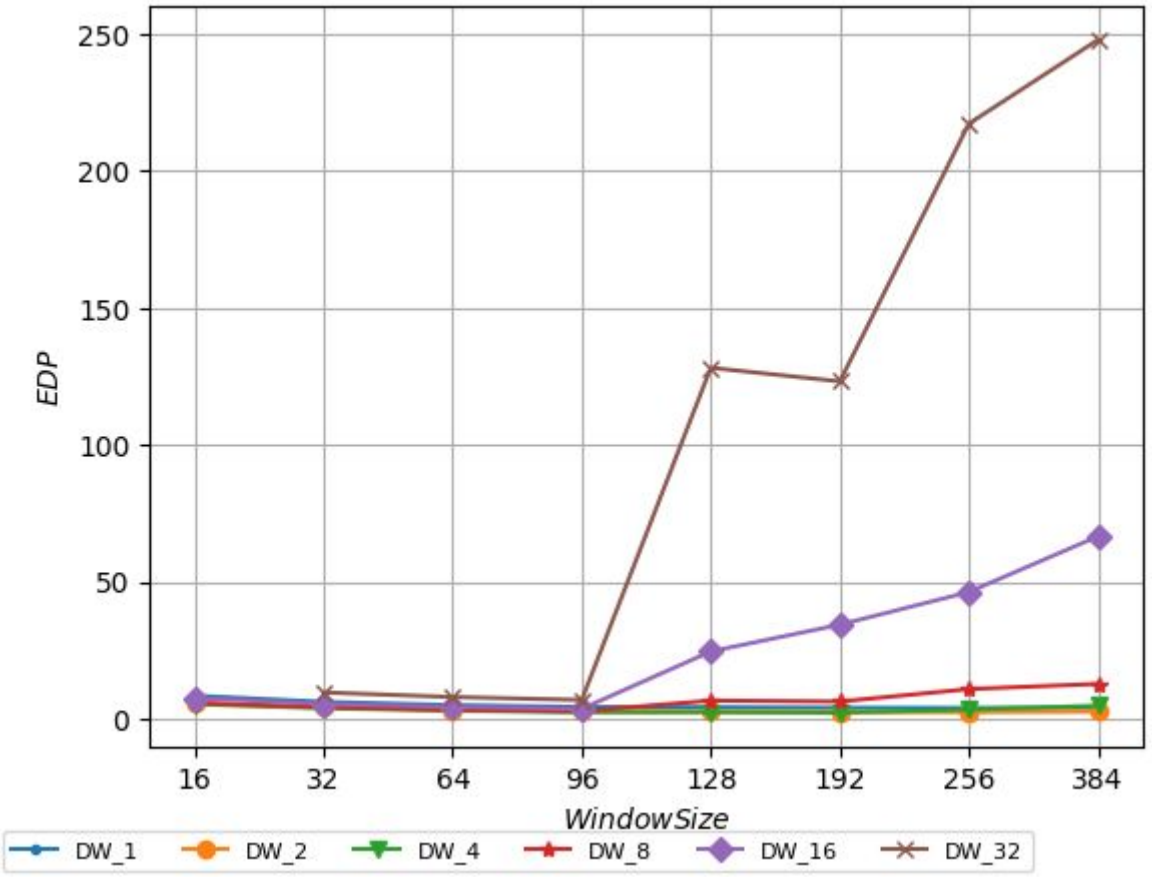
**mcf**


700
600
500
400
300
200
100
0
EDP
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

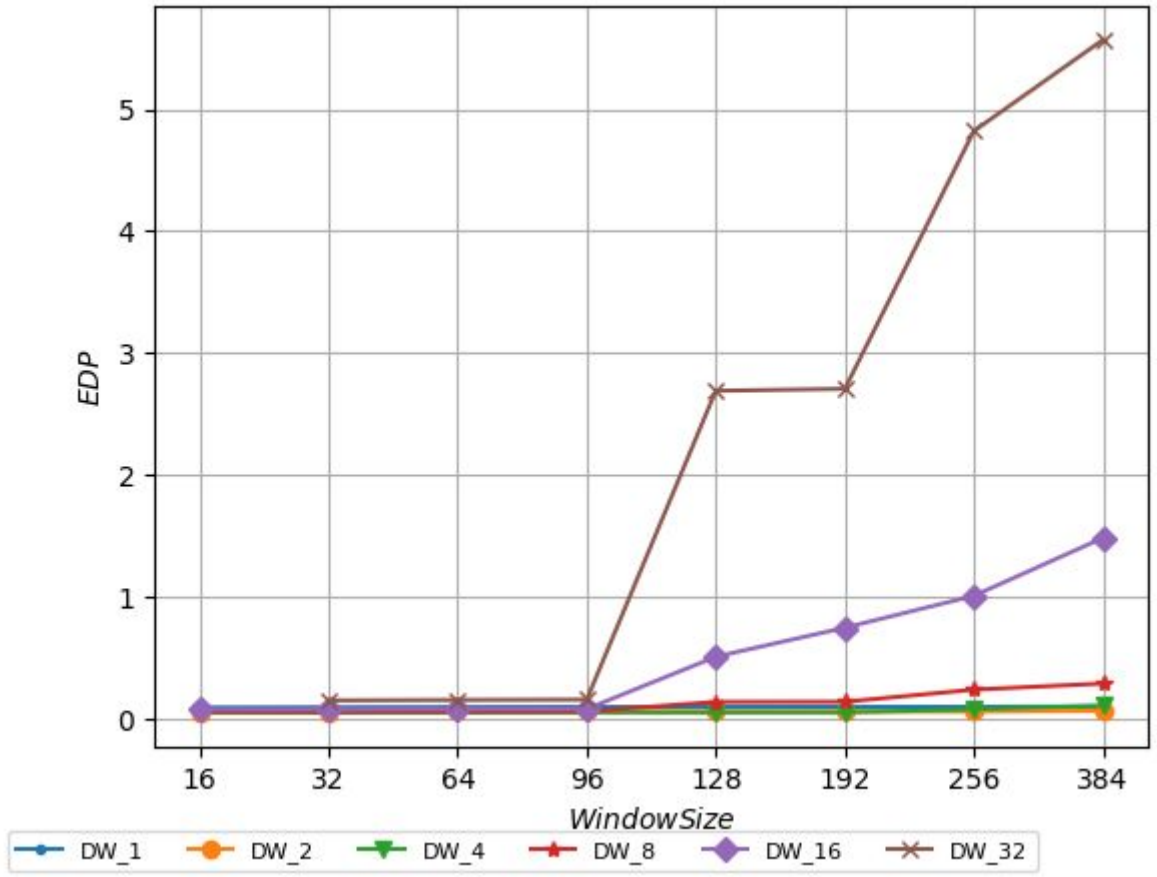
**zeusmp**


175
150
125
100
75
50
25
0
EDP
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

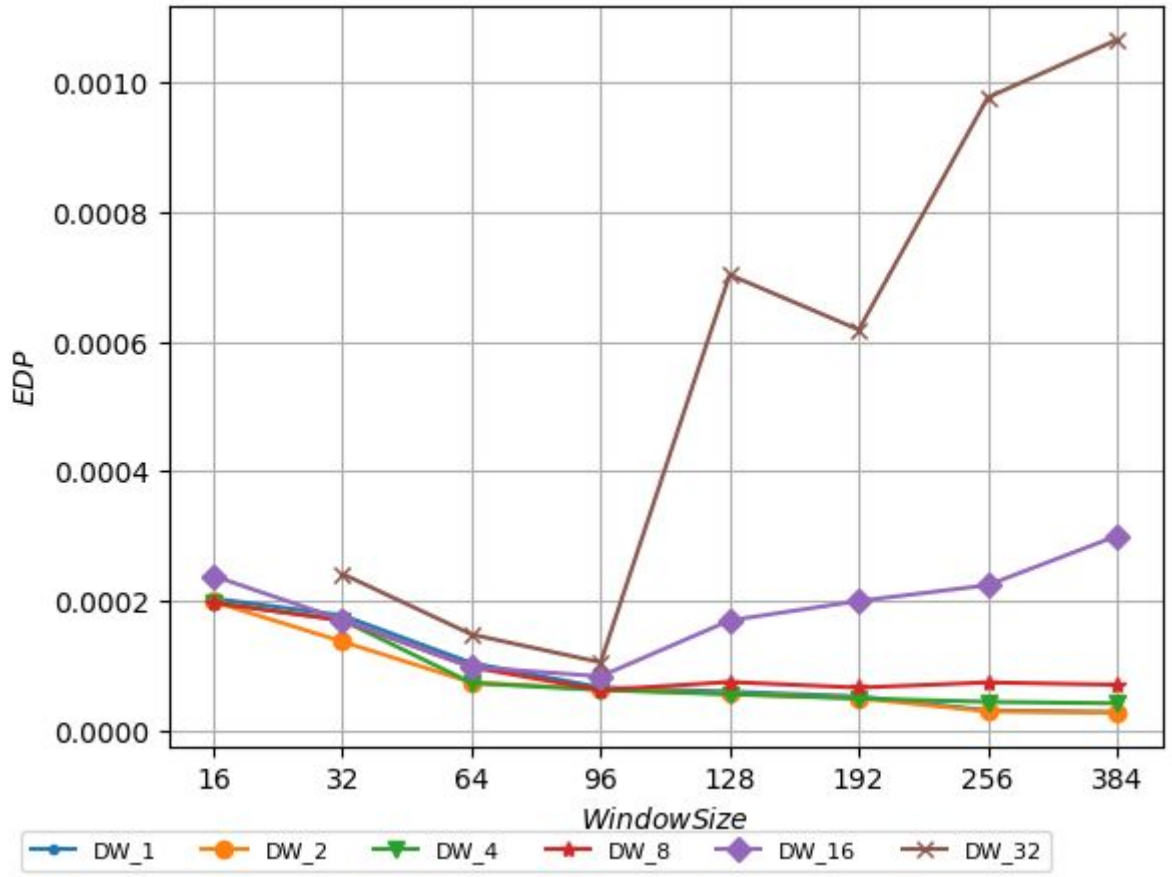
**cactusADM**


250
200
150
100
50
0
EDP
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

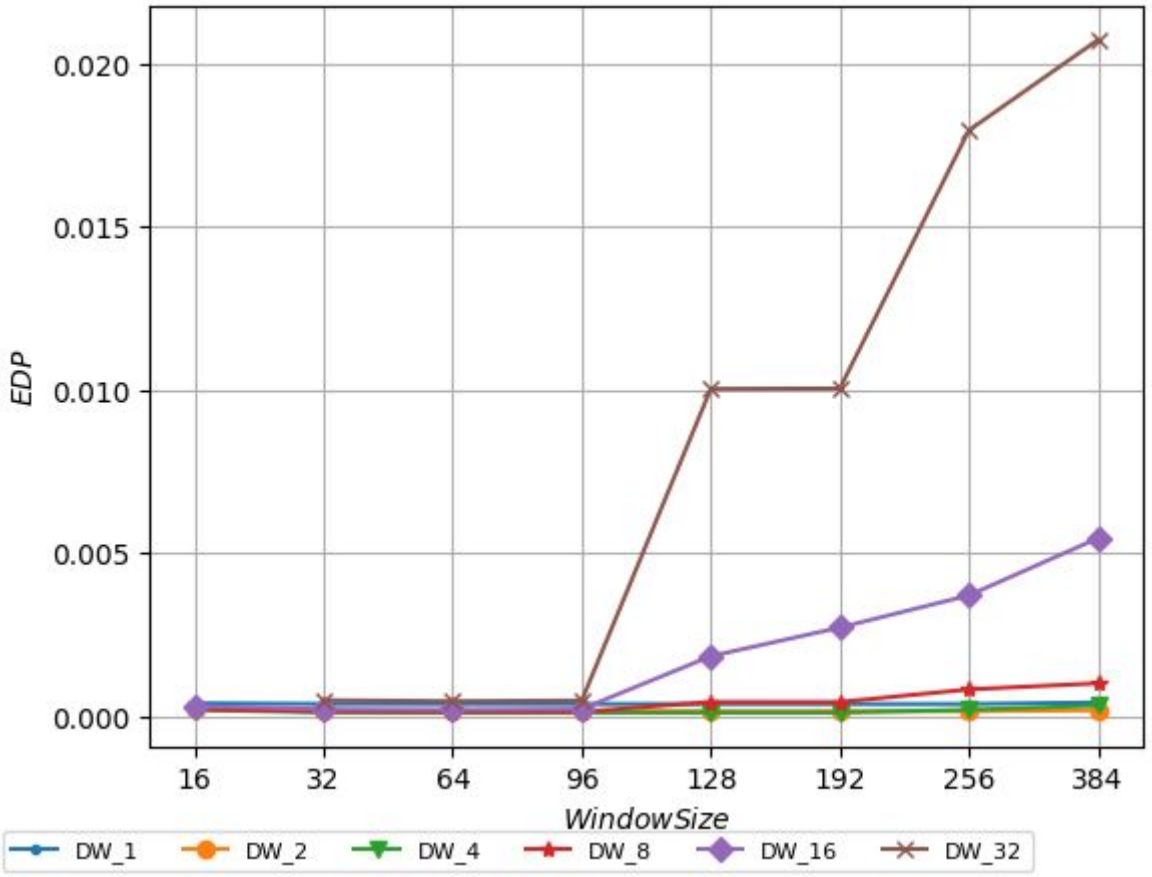
**gobmk**


EDP
0
1
2
3
4
5
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

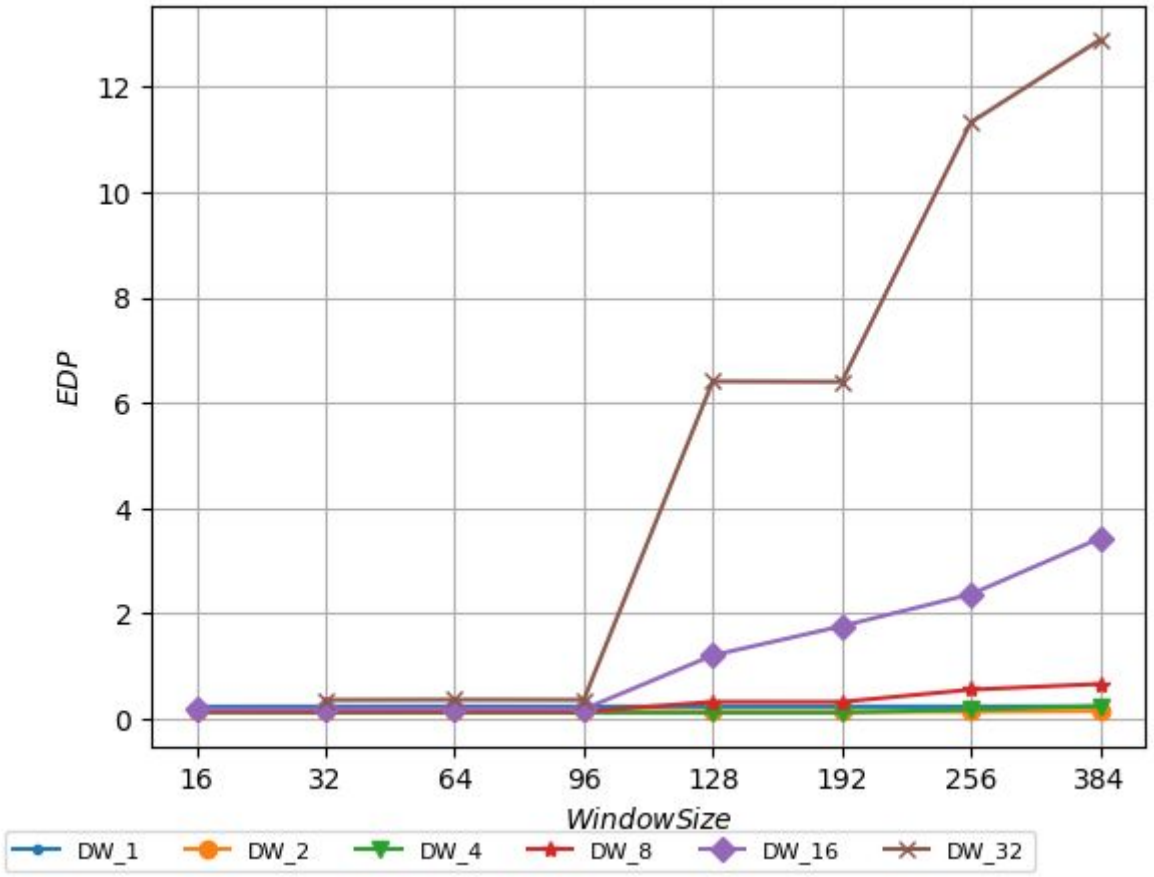
**soplex**


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
EDP
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

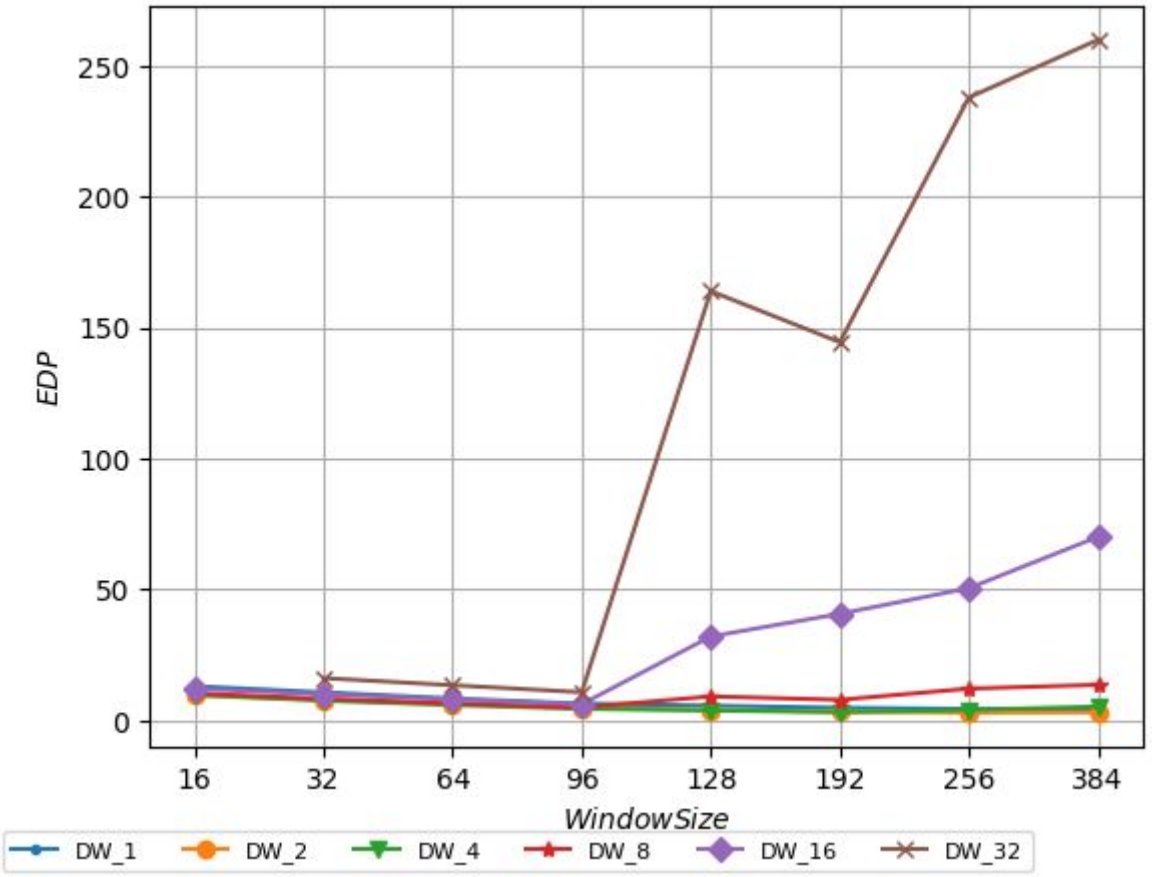
**hmmer**


0.020
0.015
0.010
0.005
0.000
EDP
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

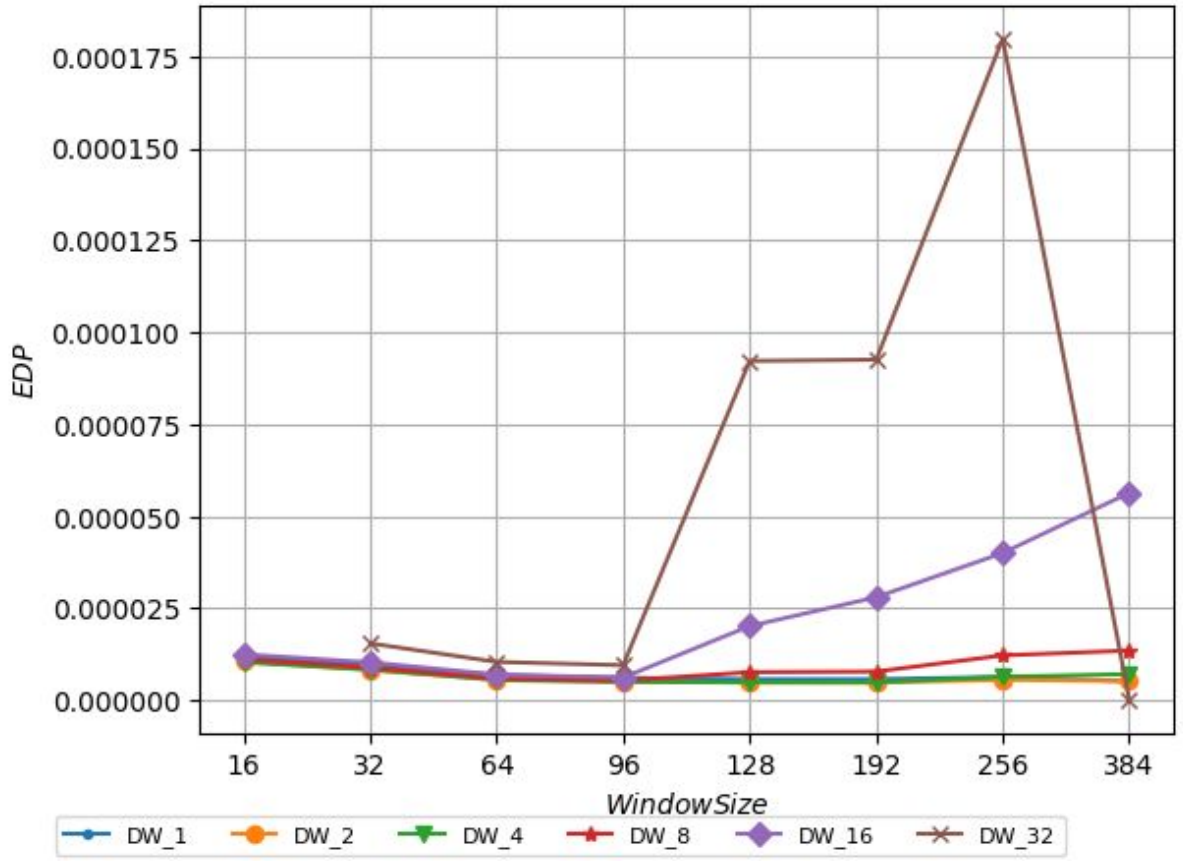
**sjeng**


12
10
8
6
4
2
0
EDP
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

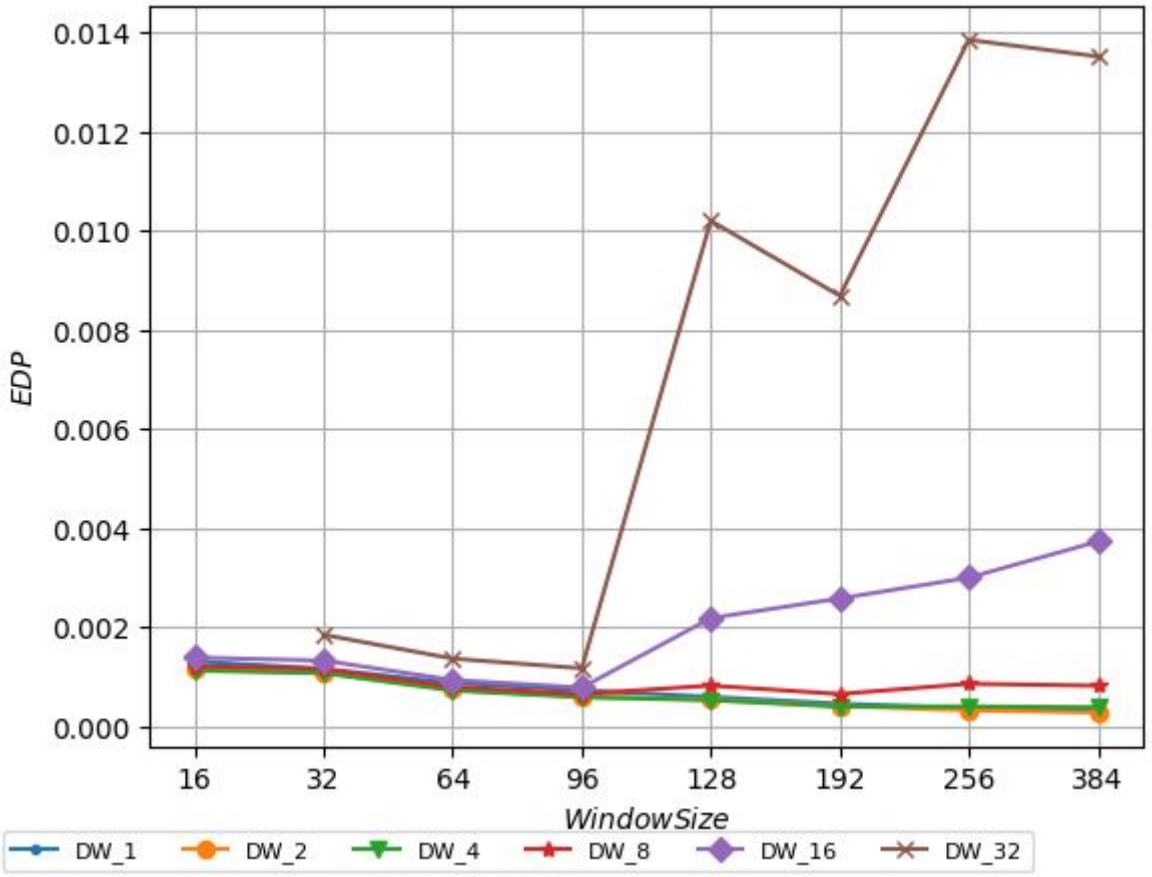
**GemsFDTD**


250
200
150
100
50
0
EDP
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

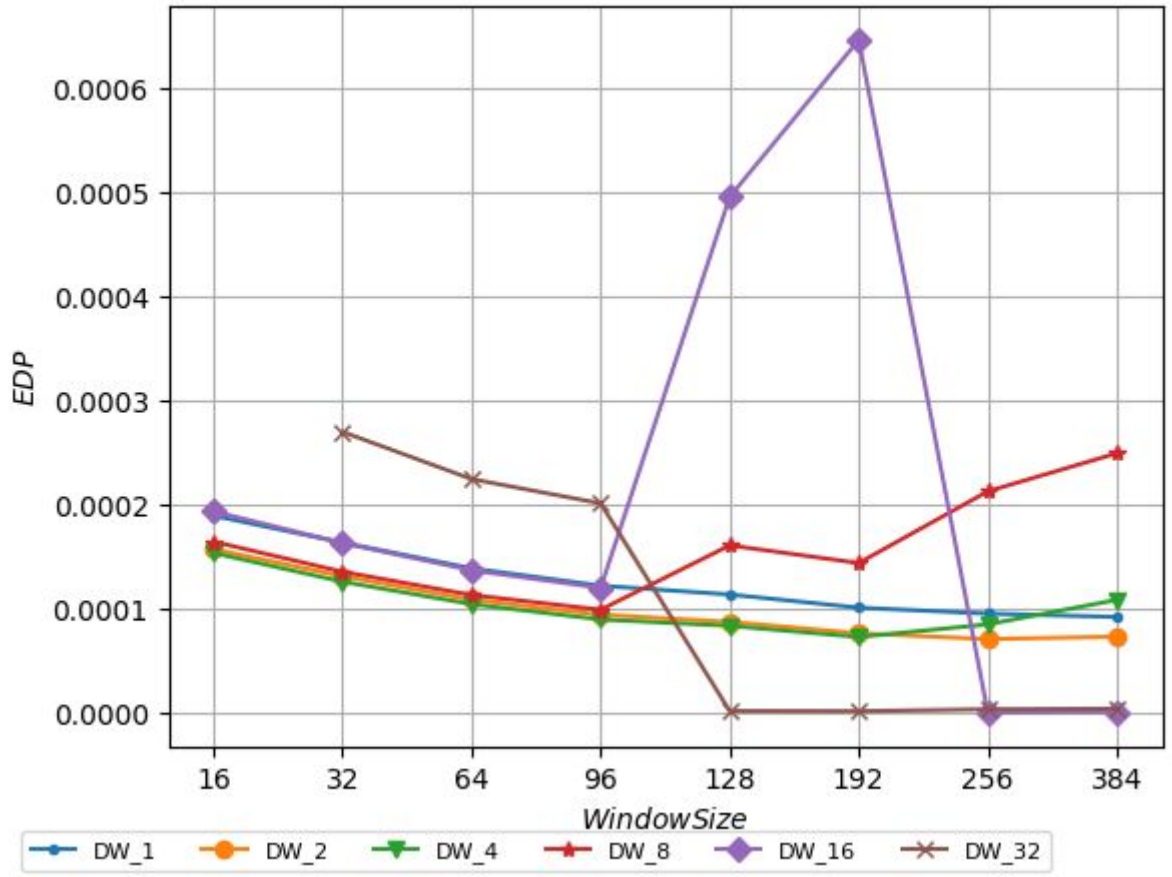
**omnetpp**


0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
EDP
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**astar**

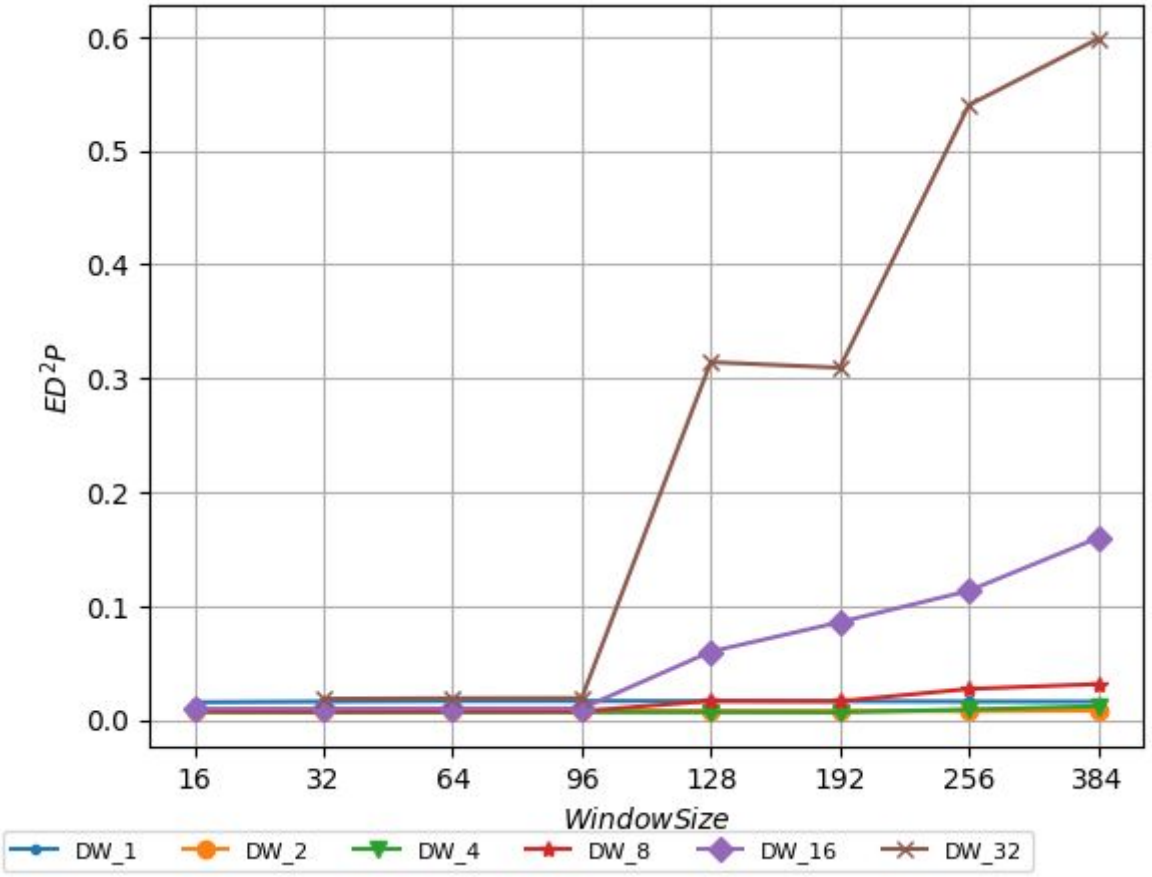

0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
EDP
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**xalancbmk**

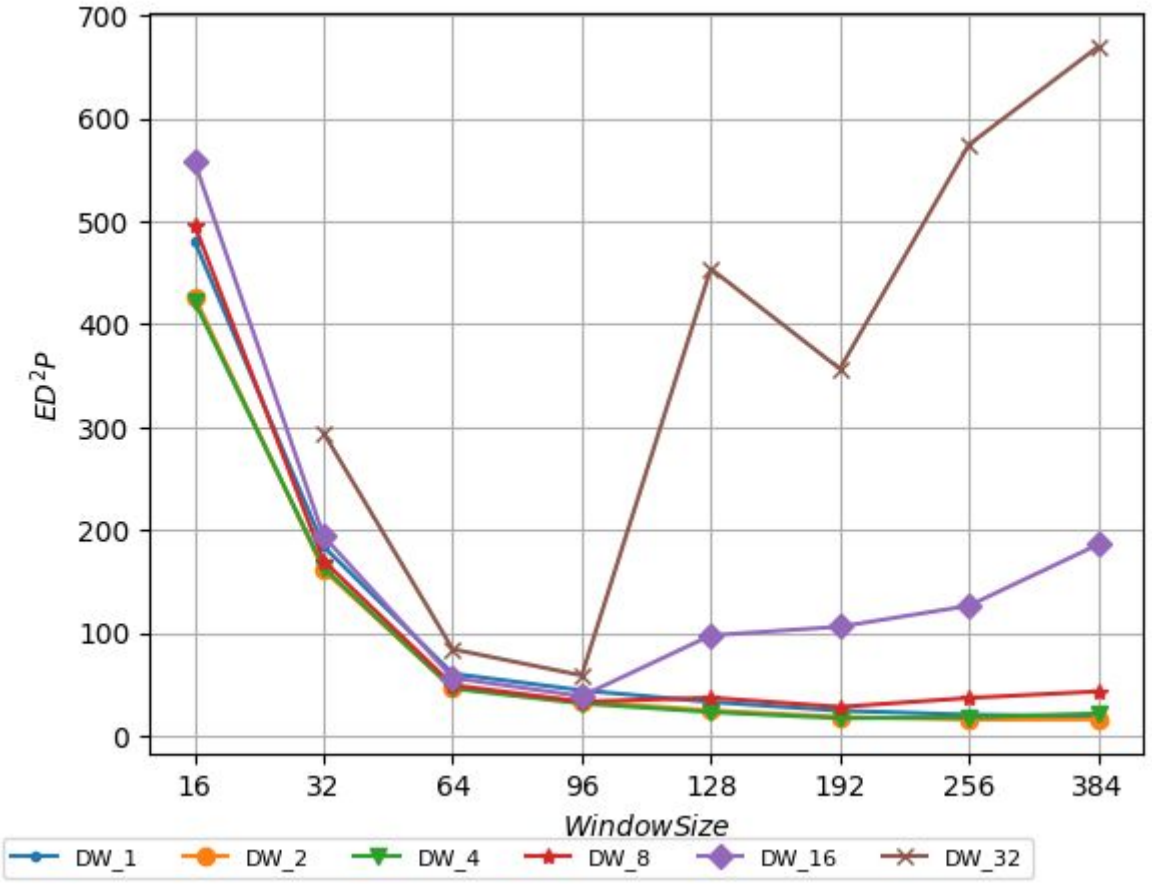

0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
EDP
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**Για το ED^2P:**

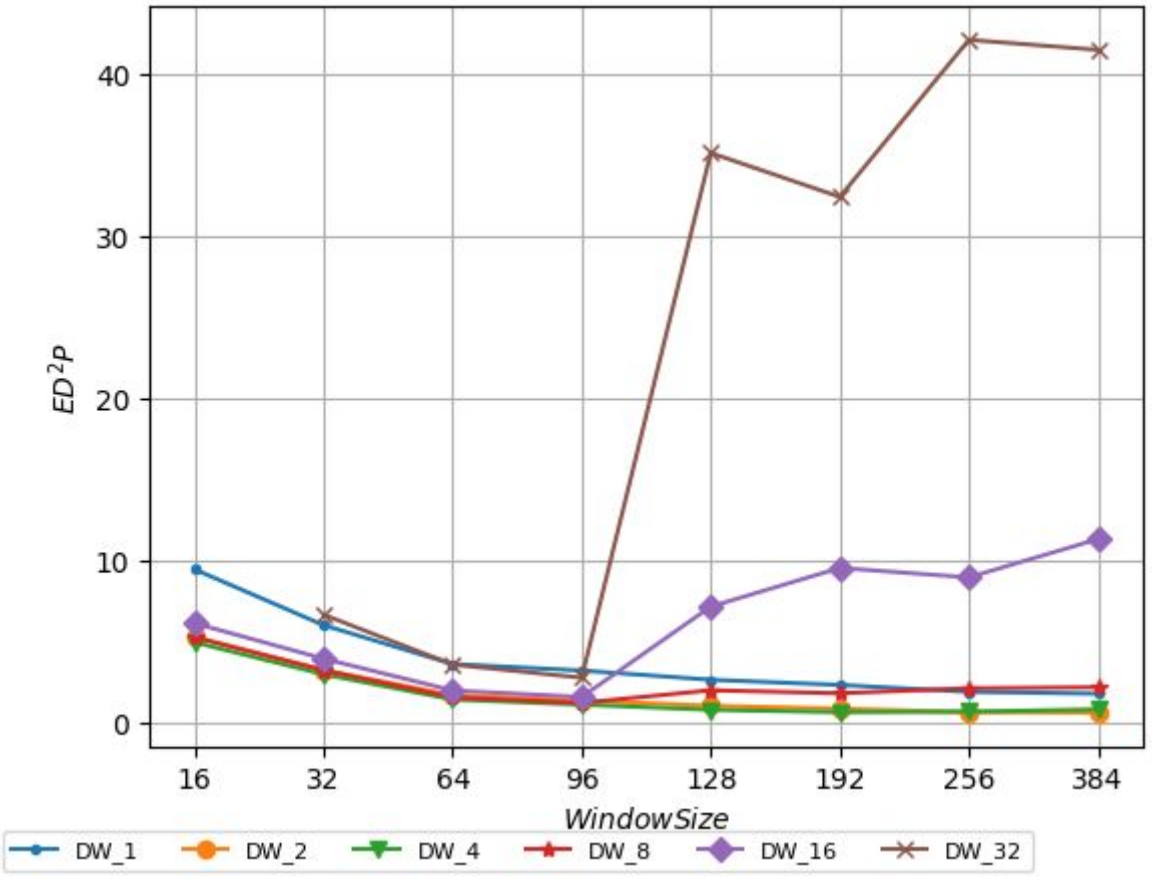
**gcc**



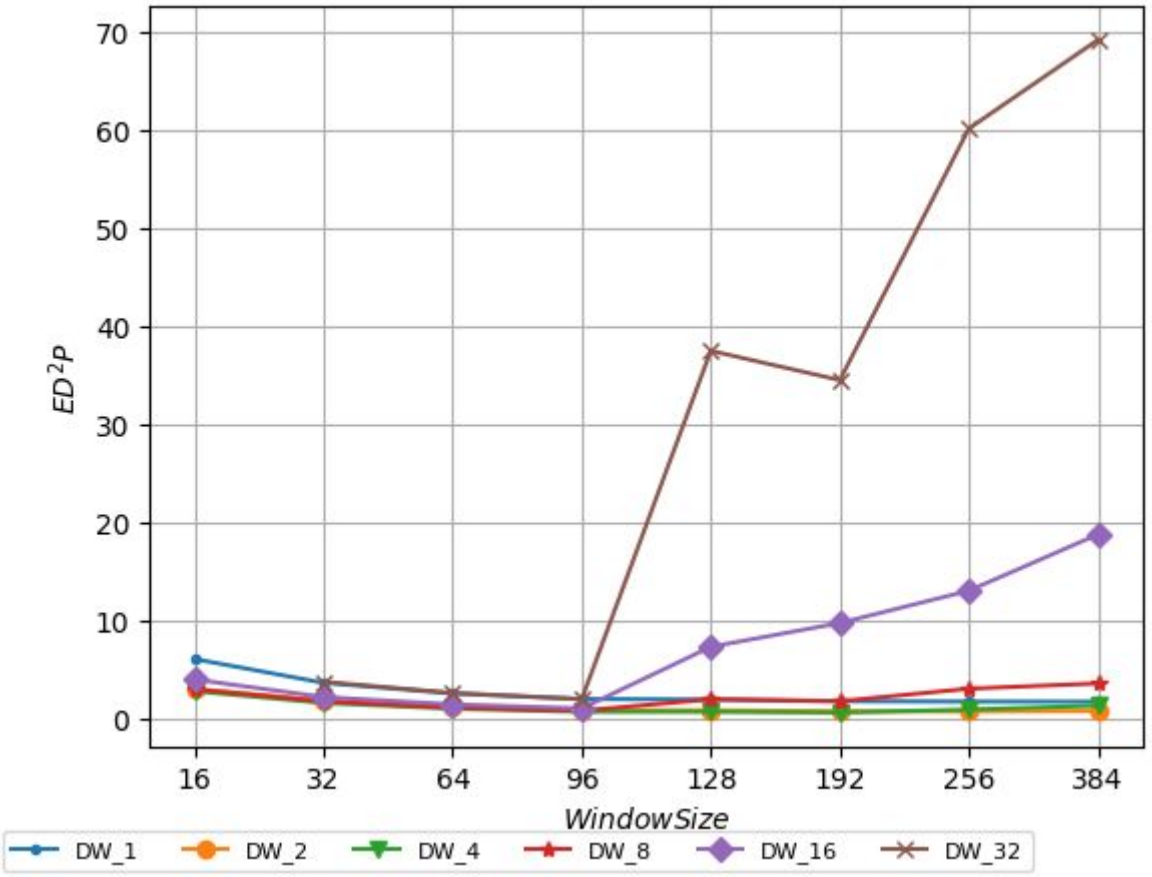
**mcf**


700
600
500
400
300
200
100
0
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

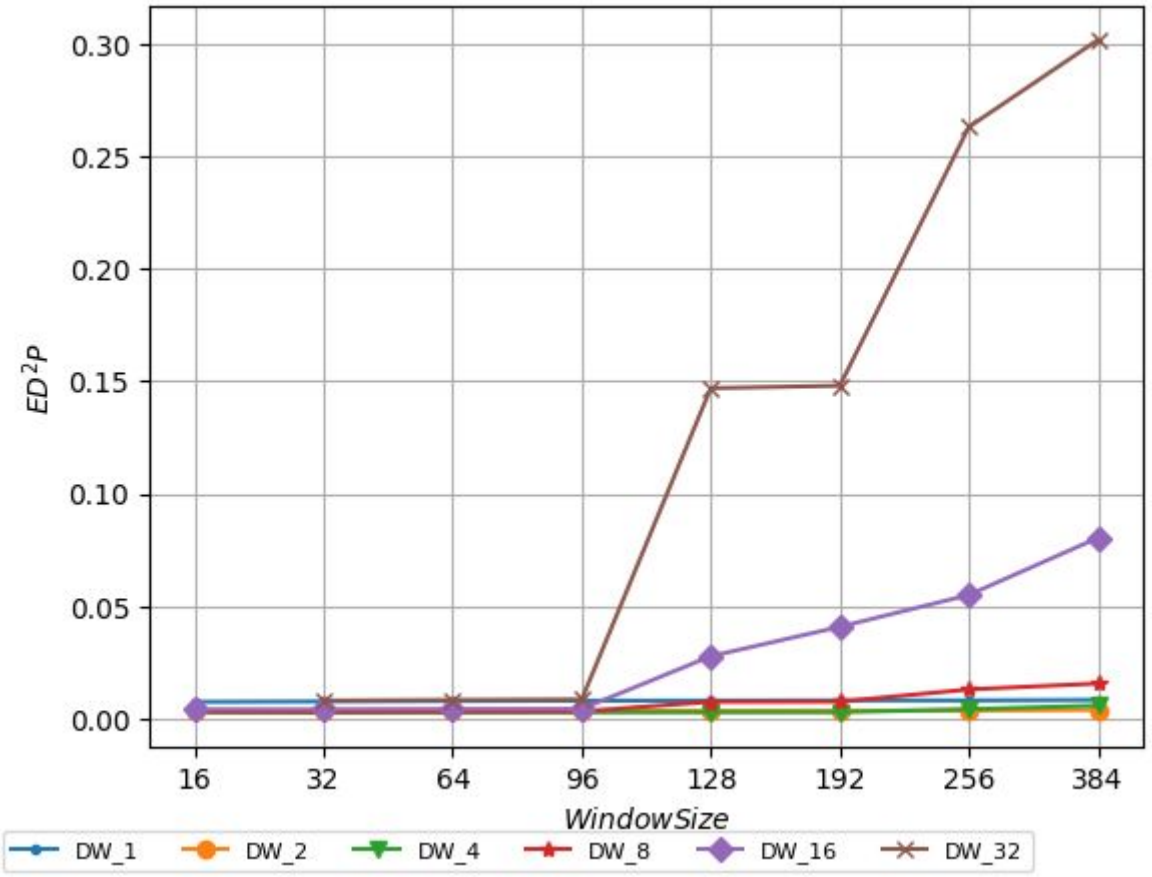
**zeusmp**


40
30
20
10
0
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

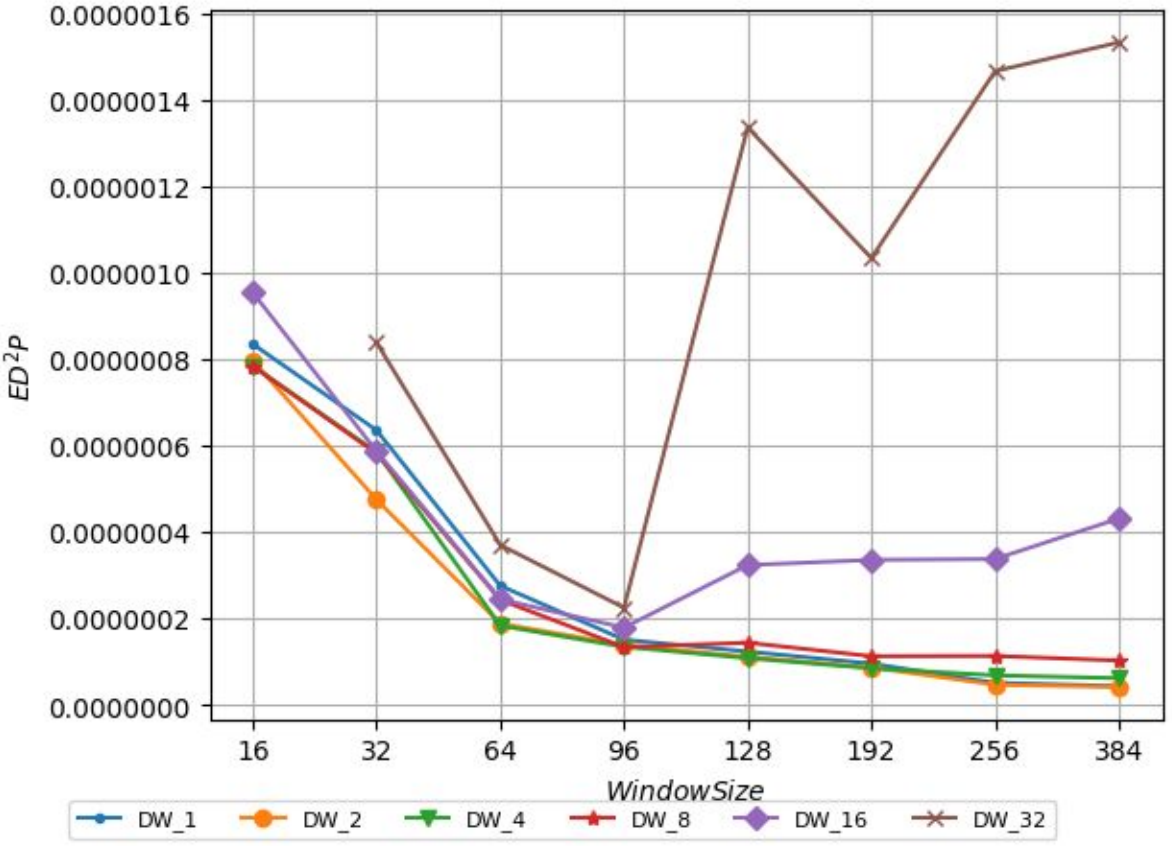
**cactusADM**


70
60
50
40
30
20
10
0
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

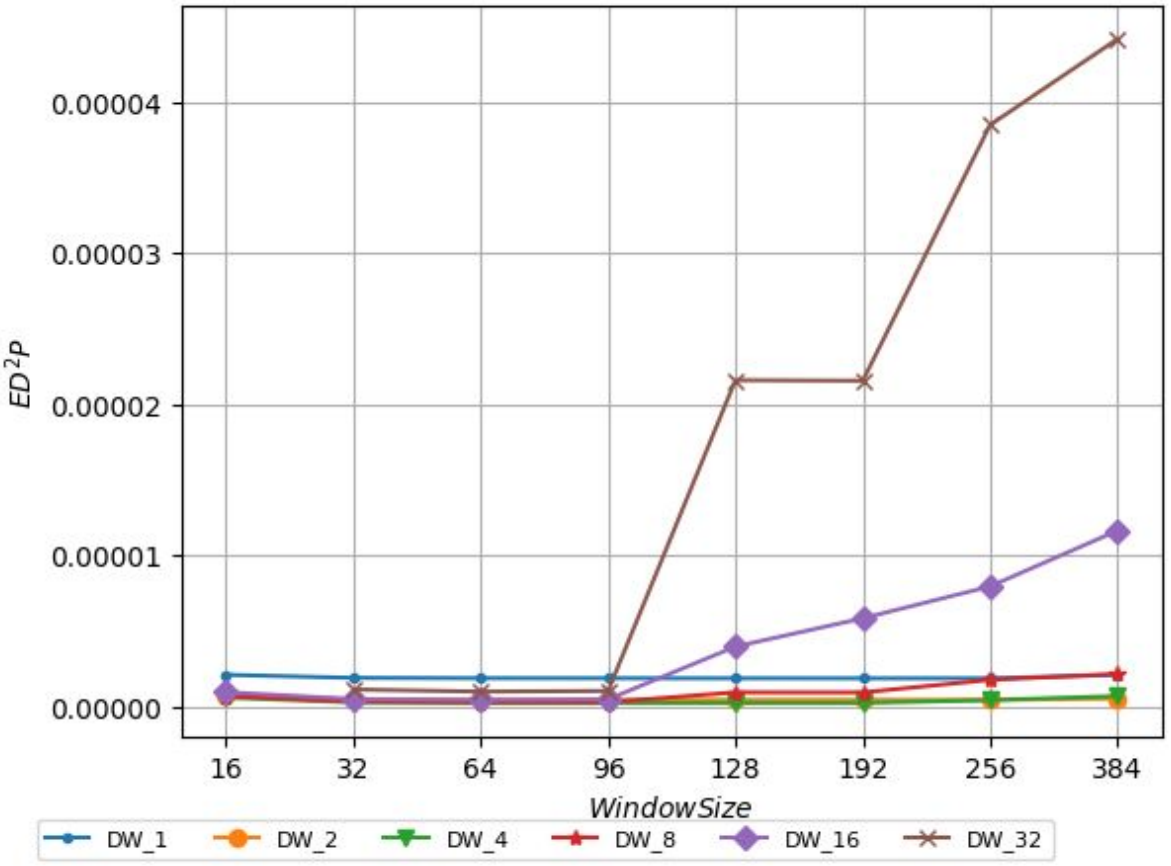
**gobmk**


0.30
0.25
0.20
0.15
0.10
0.05
0.00
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

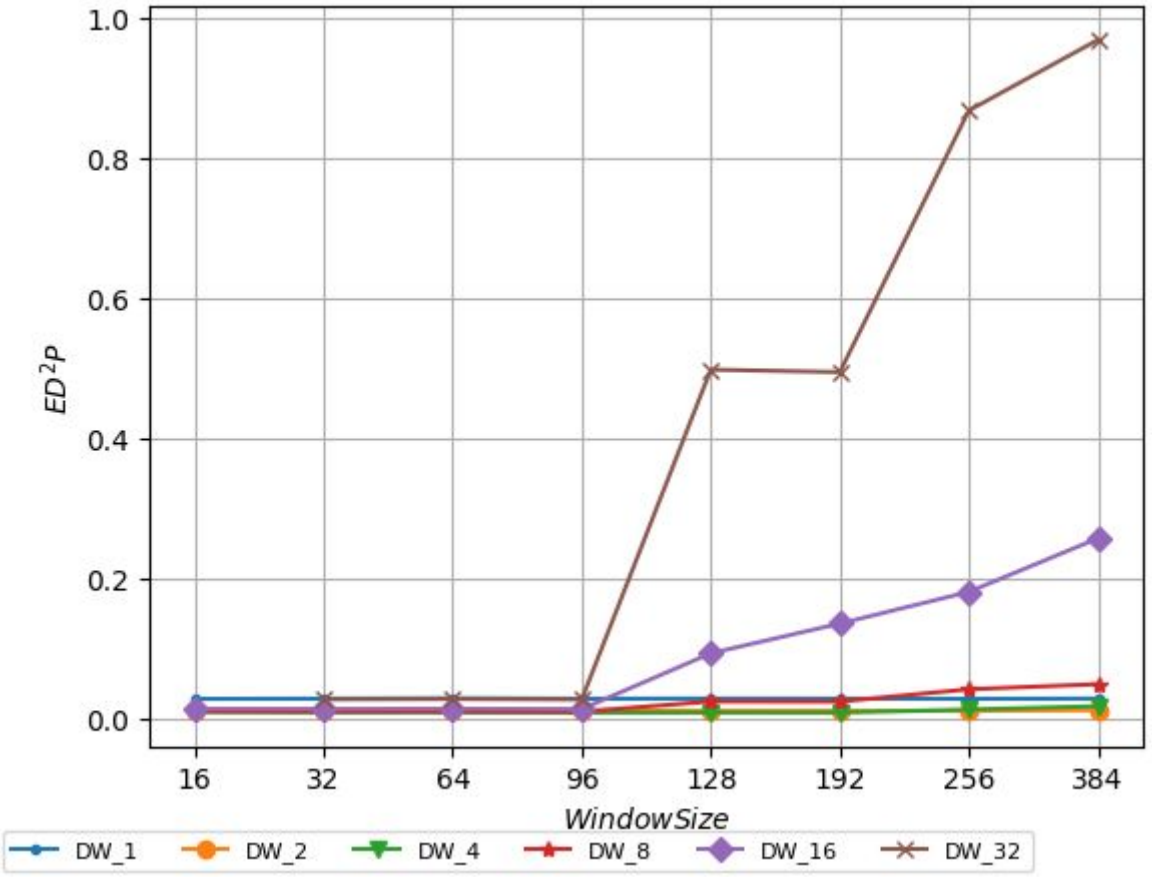
**soplex**


0.0000016
0.0000014
0.0000012
0.0000010
0.0000008
0.0000006
0.0000004
0.0000002
0.0000000
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

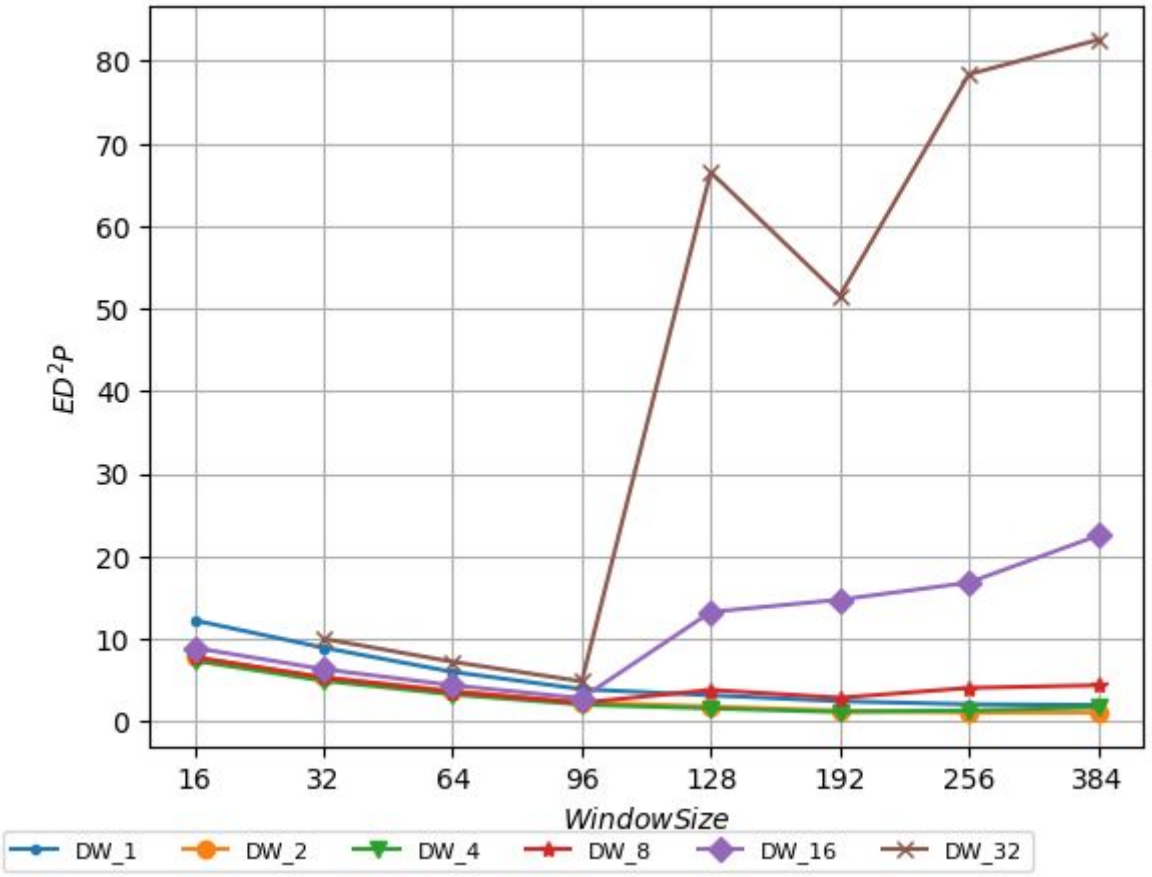
**hmmer**


0.00004
0.00003
0.00002
0.00001
0.00000
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

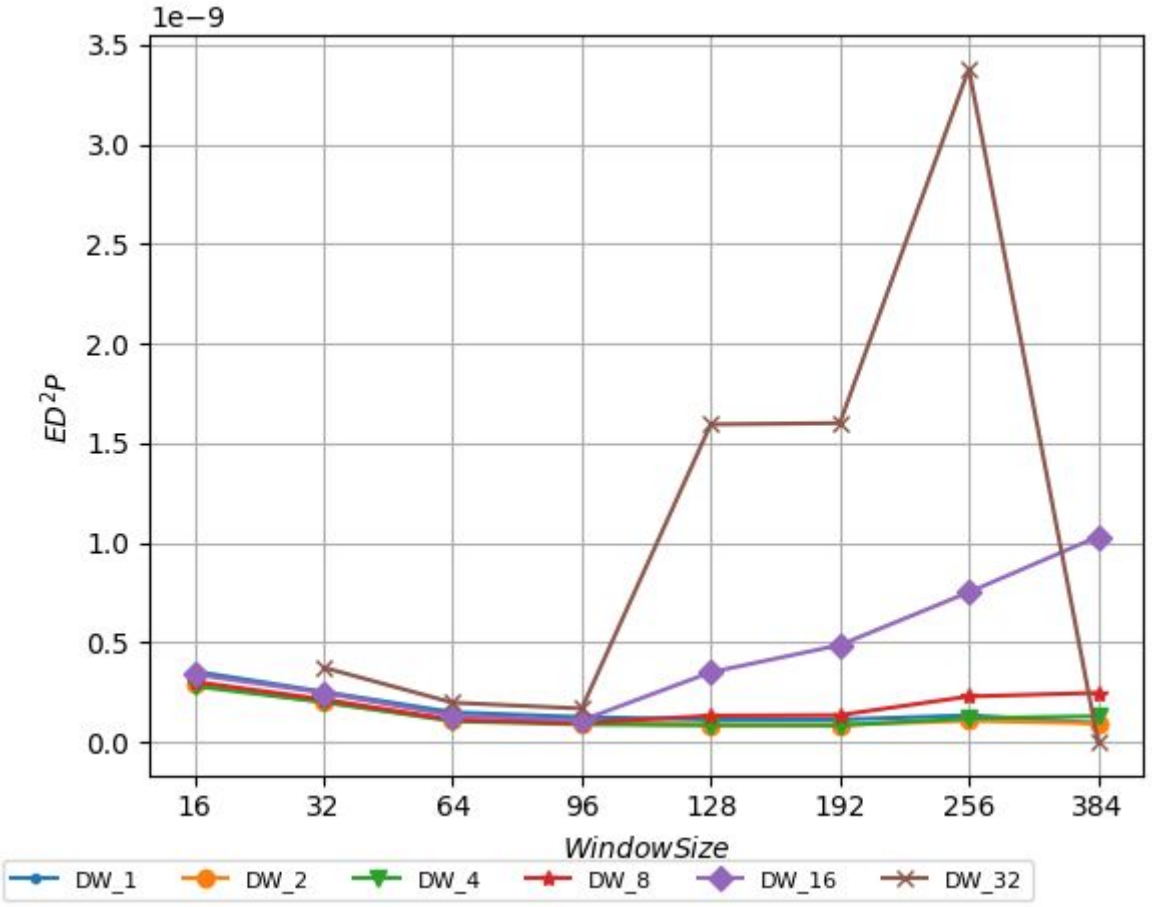
**sjeng**


1.0
0.8
0.6
0.4
0.2
0.0
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

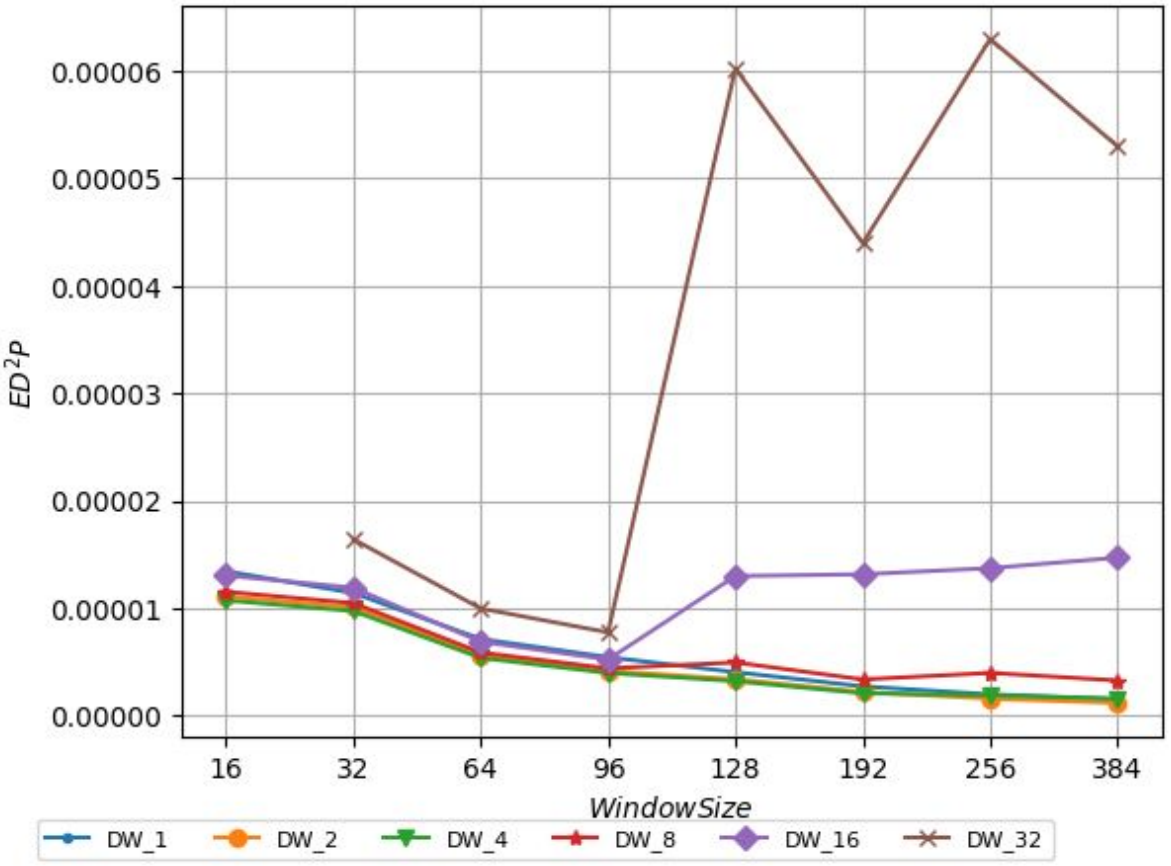
**GemsFDTD**


80
70
60
50
40
30
20
10
0
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

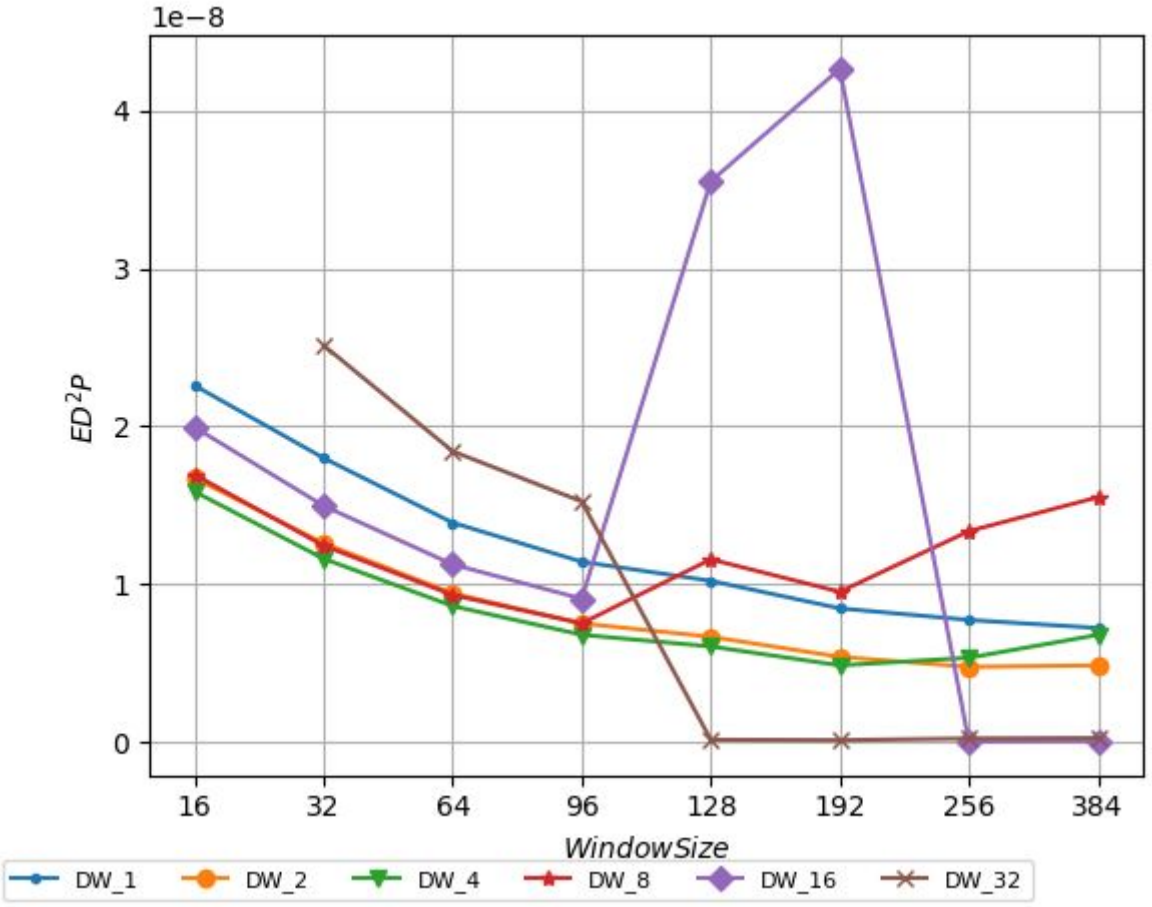
**omnetpp**


1e−9
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**astar**

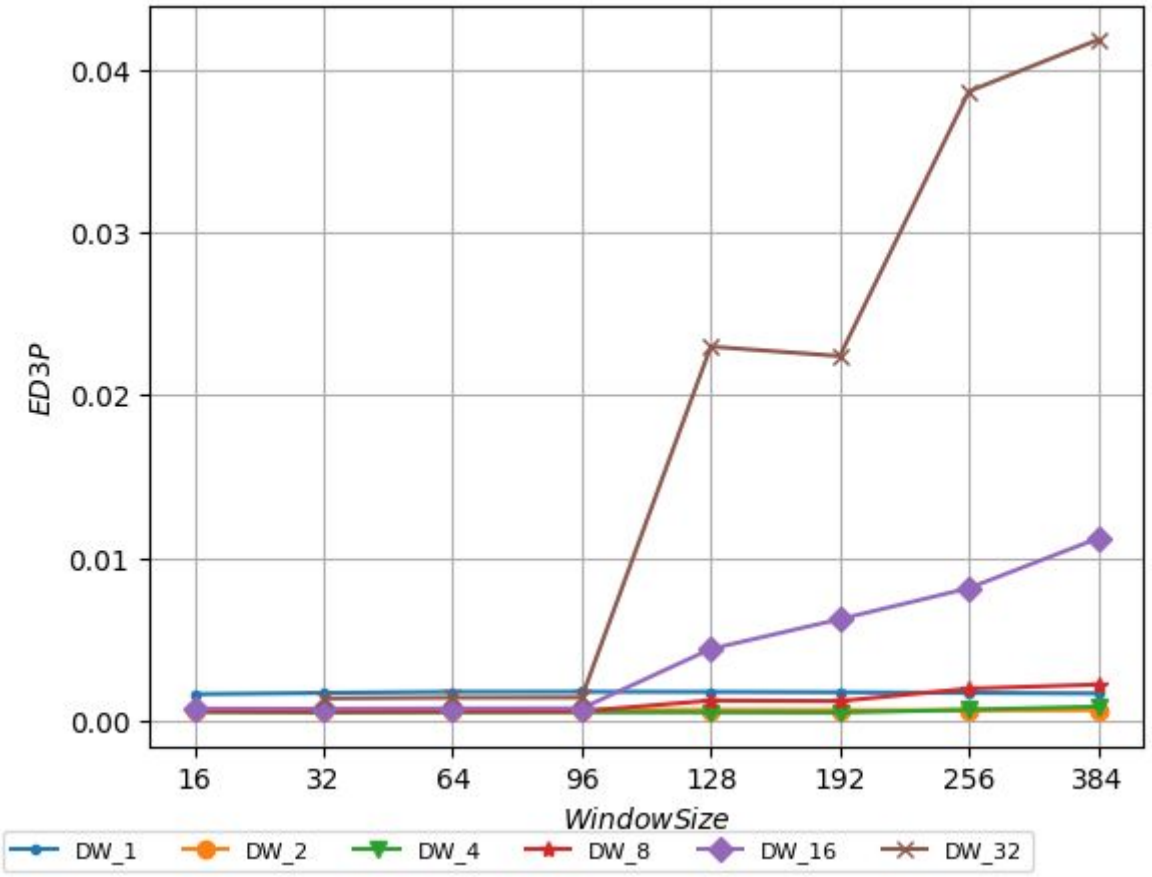

0.00006
0.00005
0.00004
0.00003
0.00002
0.00001
0.00000
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**xalancbmk**

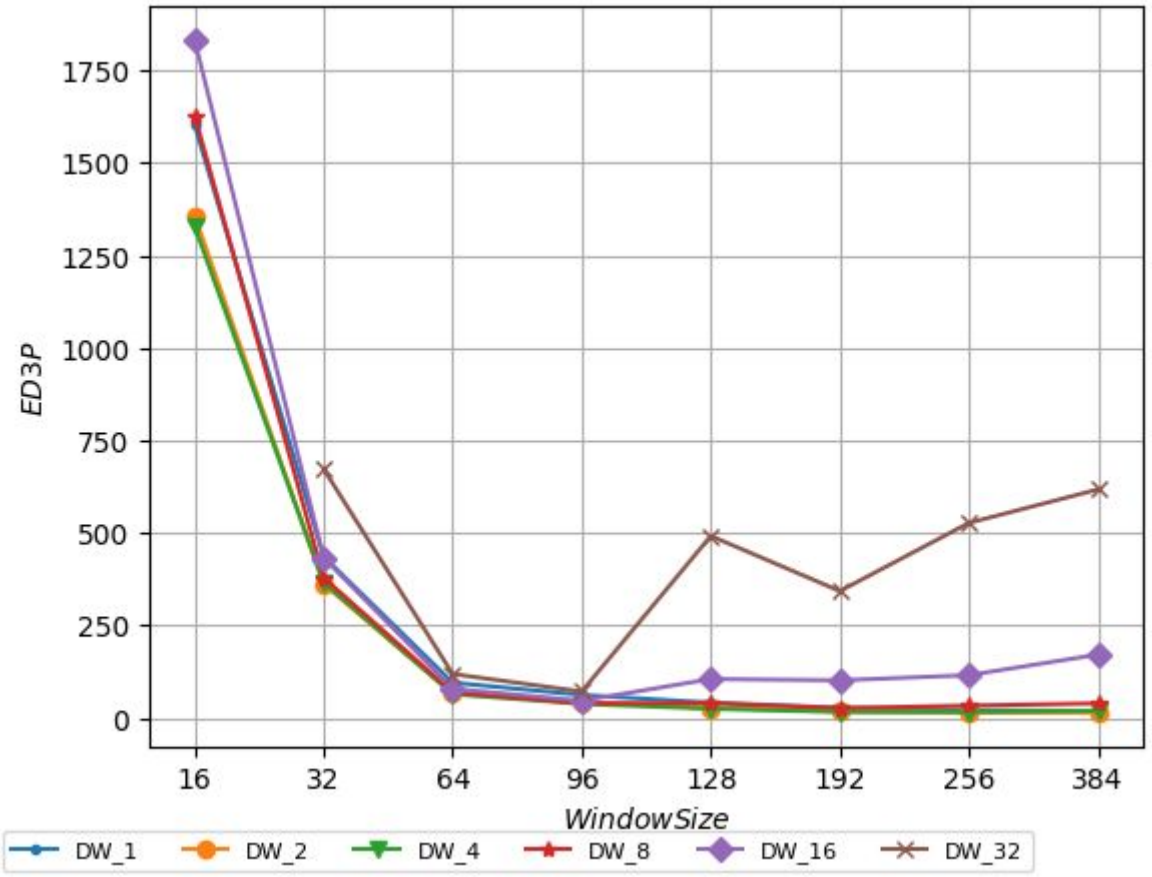

1e−8
4
3
2
1
0
$ED^2P$
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**Για το ED^3P:**

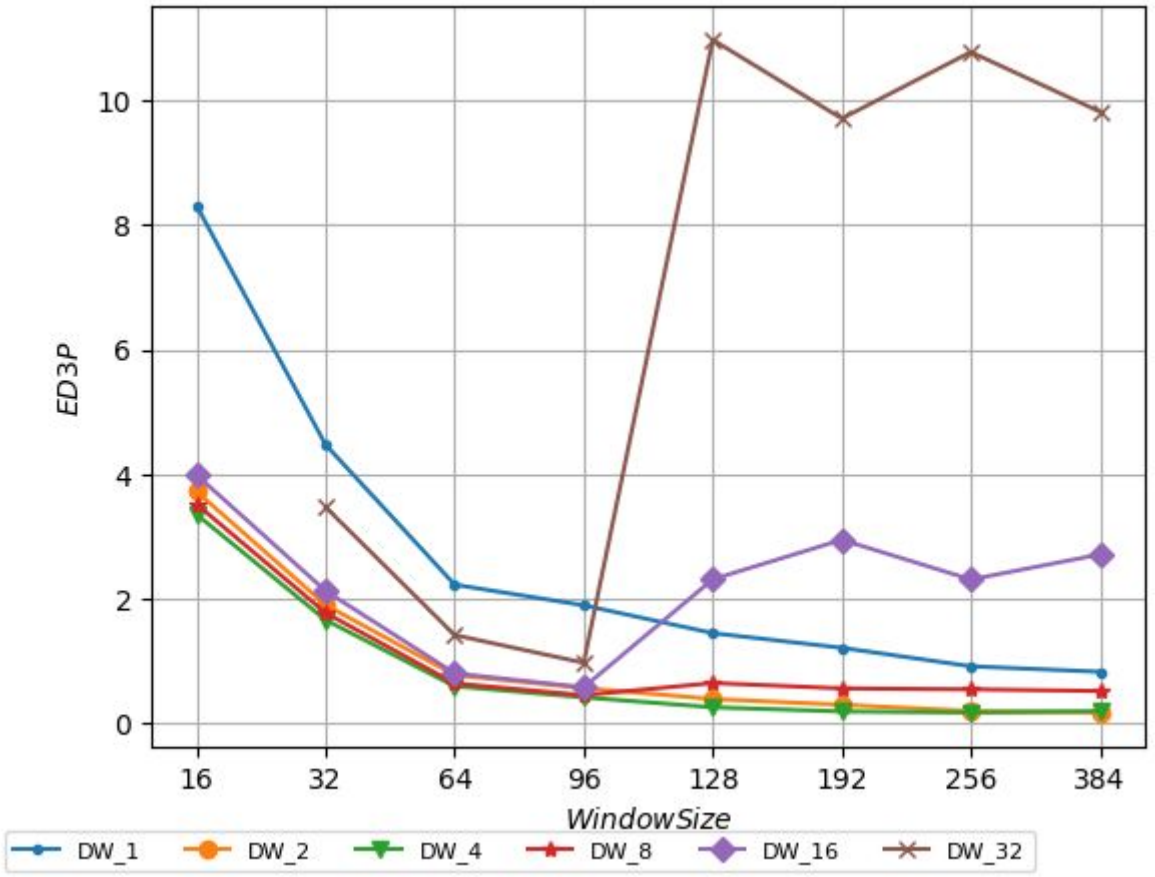
**gcc**



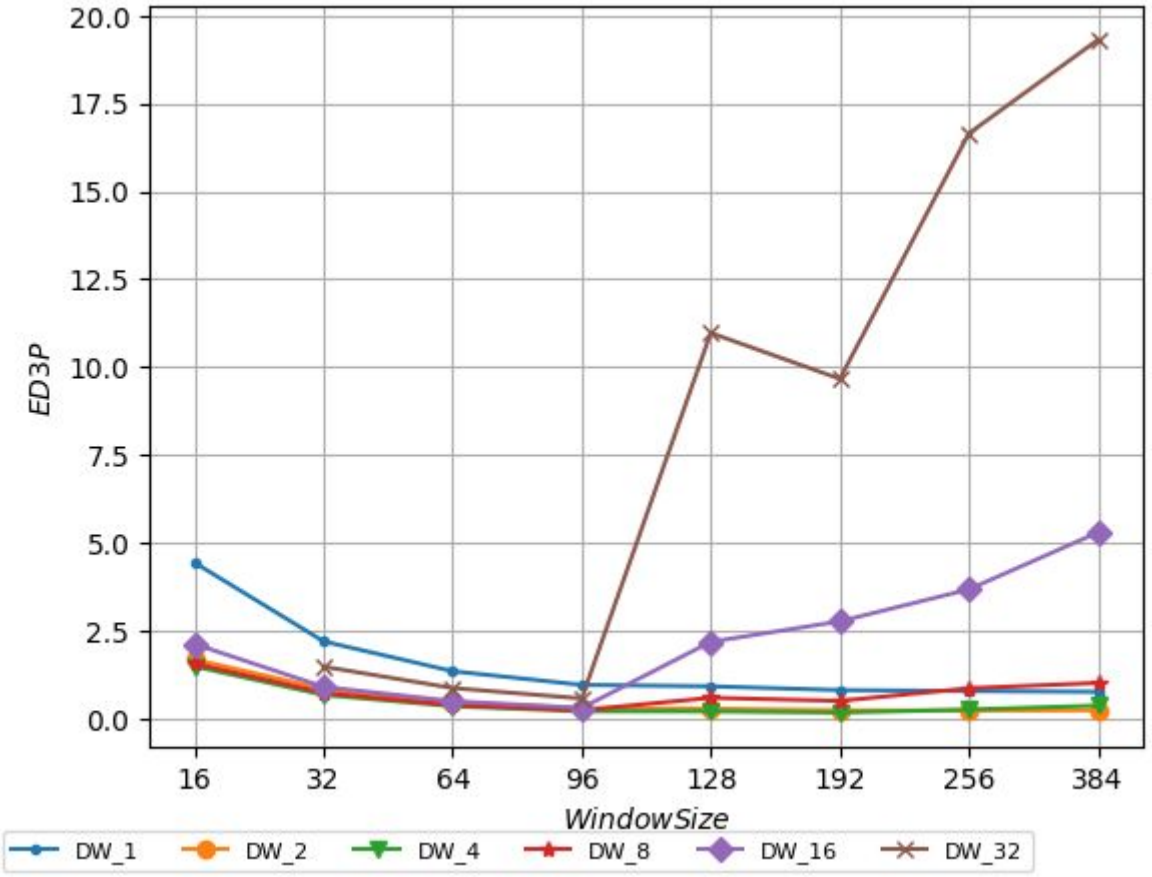
**mcf**


1750
1500
1250
1000
750
500
250
0
ED3P
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

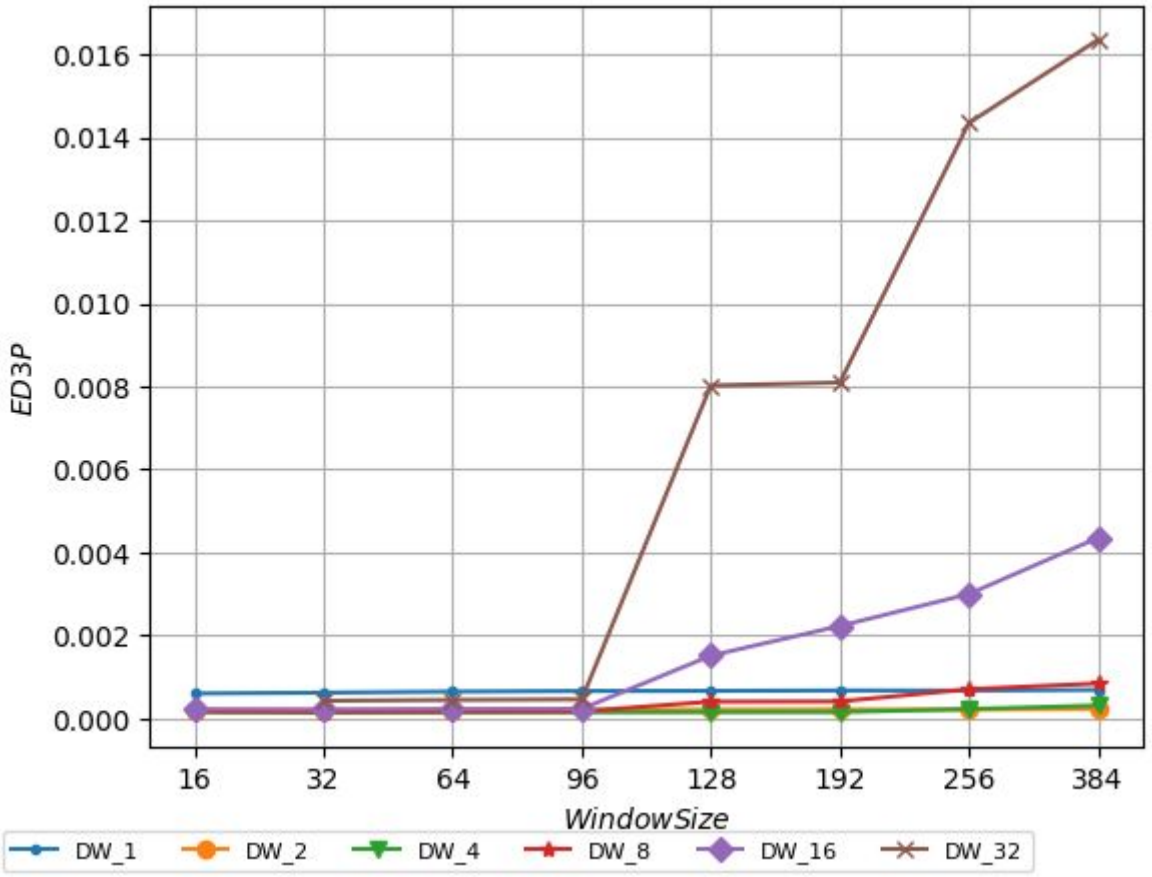
**zeusmp**


ED3P
0
2
4
6
8
10
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

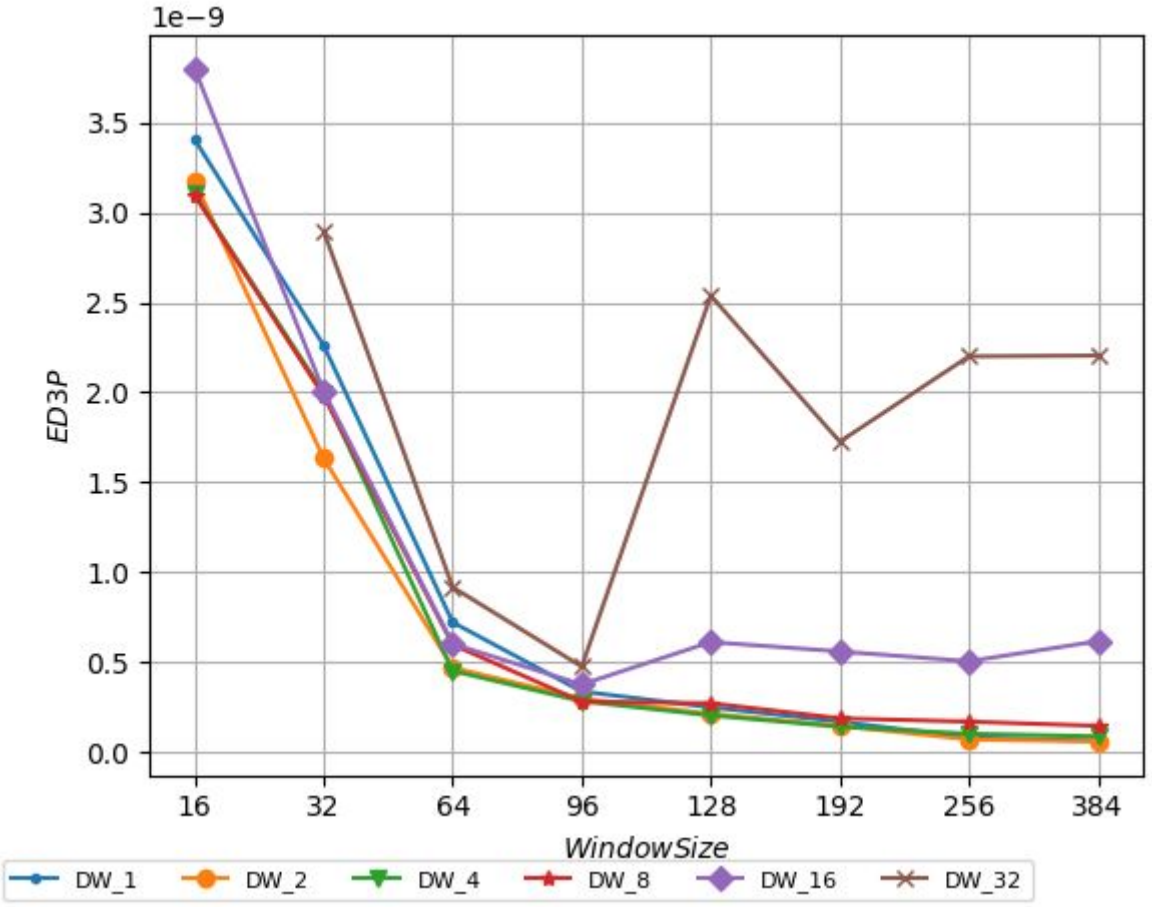
**cactusADM**


20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
ED3P
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

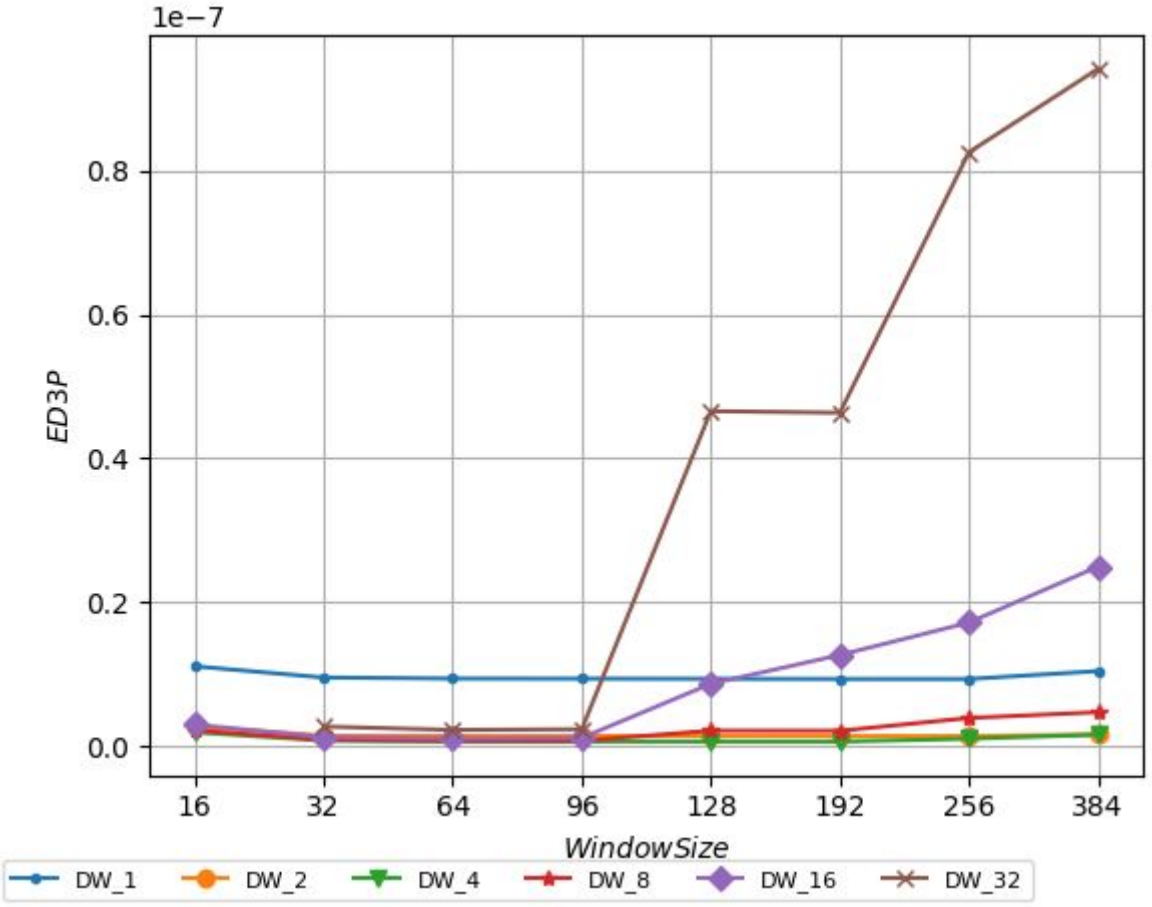
**gobmk**


0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
ED3P
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

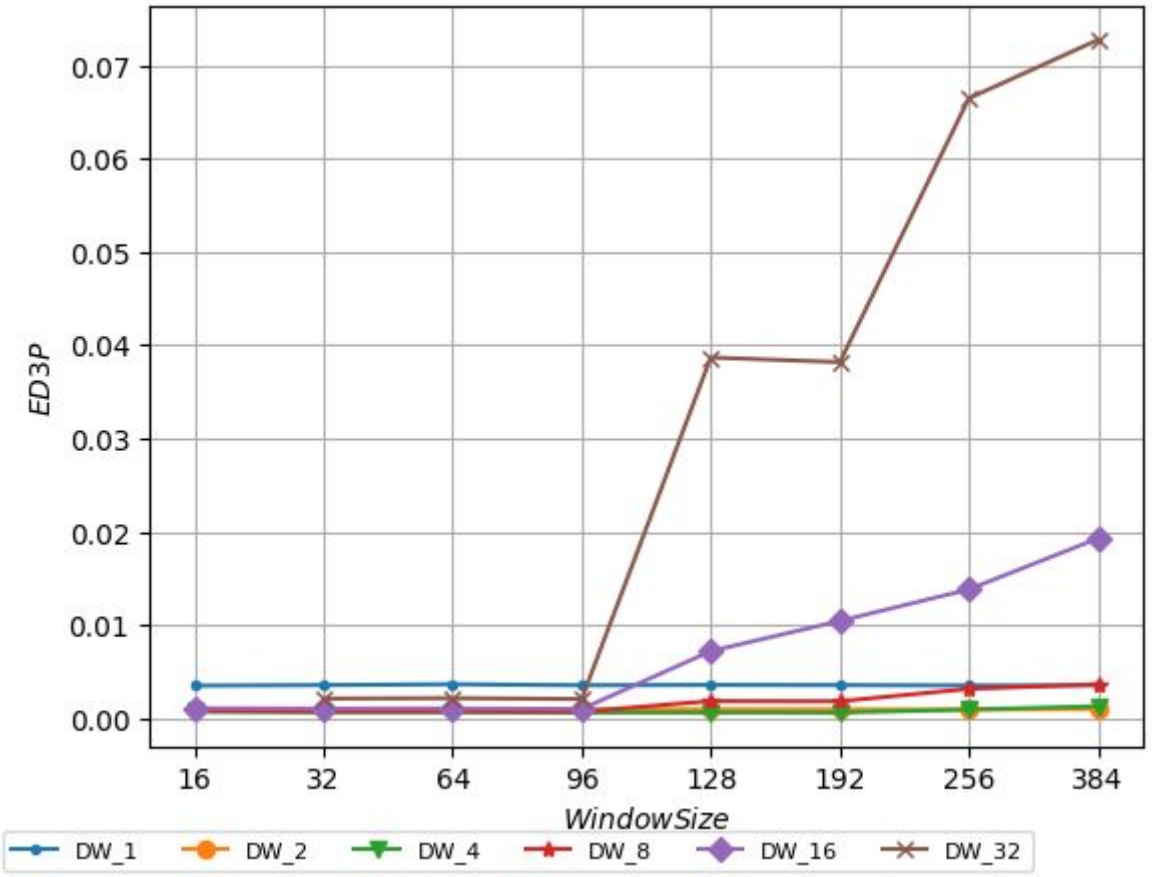
**soplex**


1e−9
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
ED3P
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

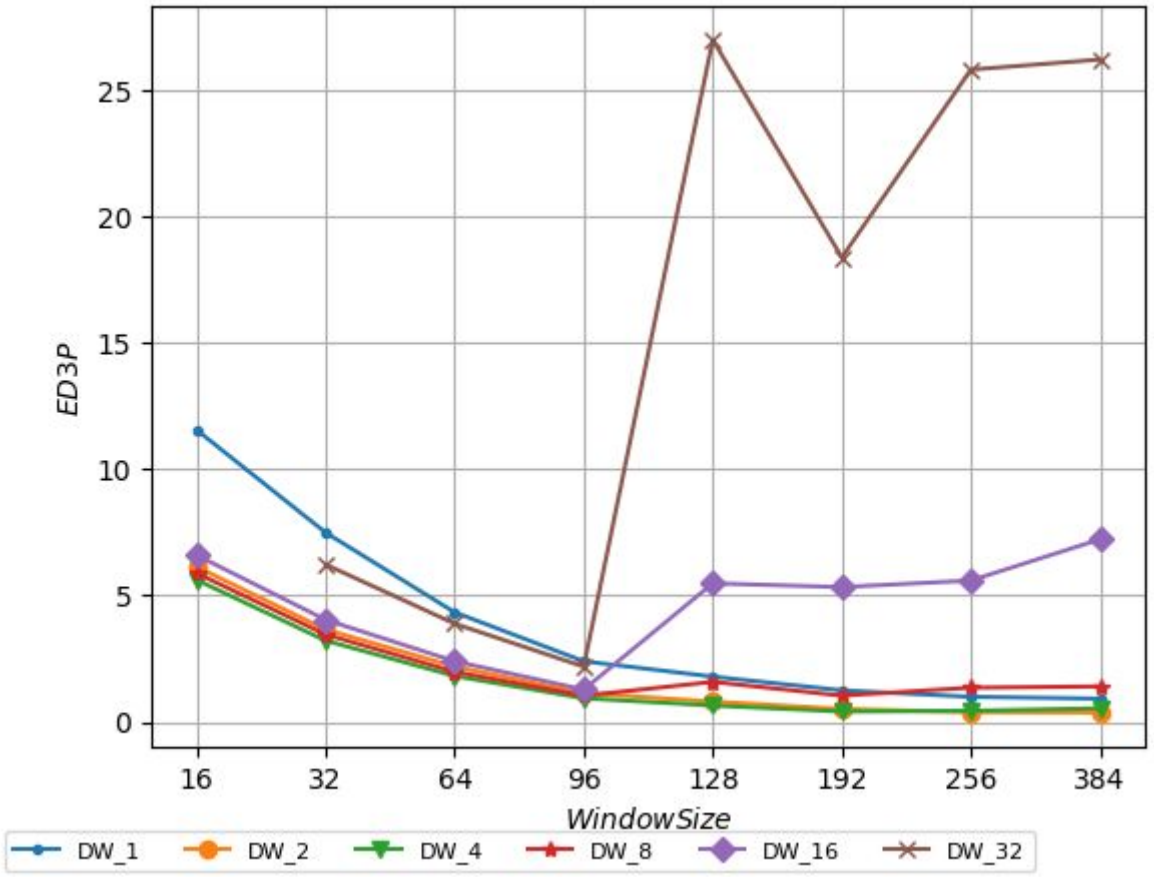
**hmmer**


1e−7
0.8
0.6
0.4
0.2
0.0
ED3P
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

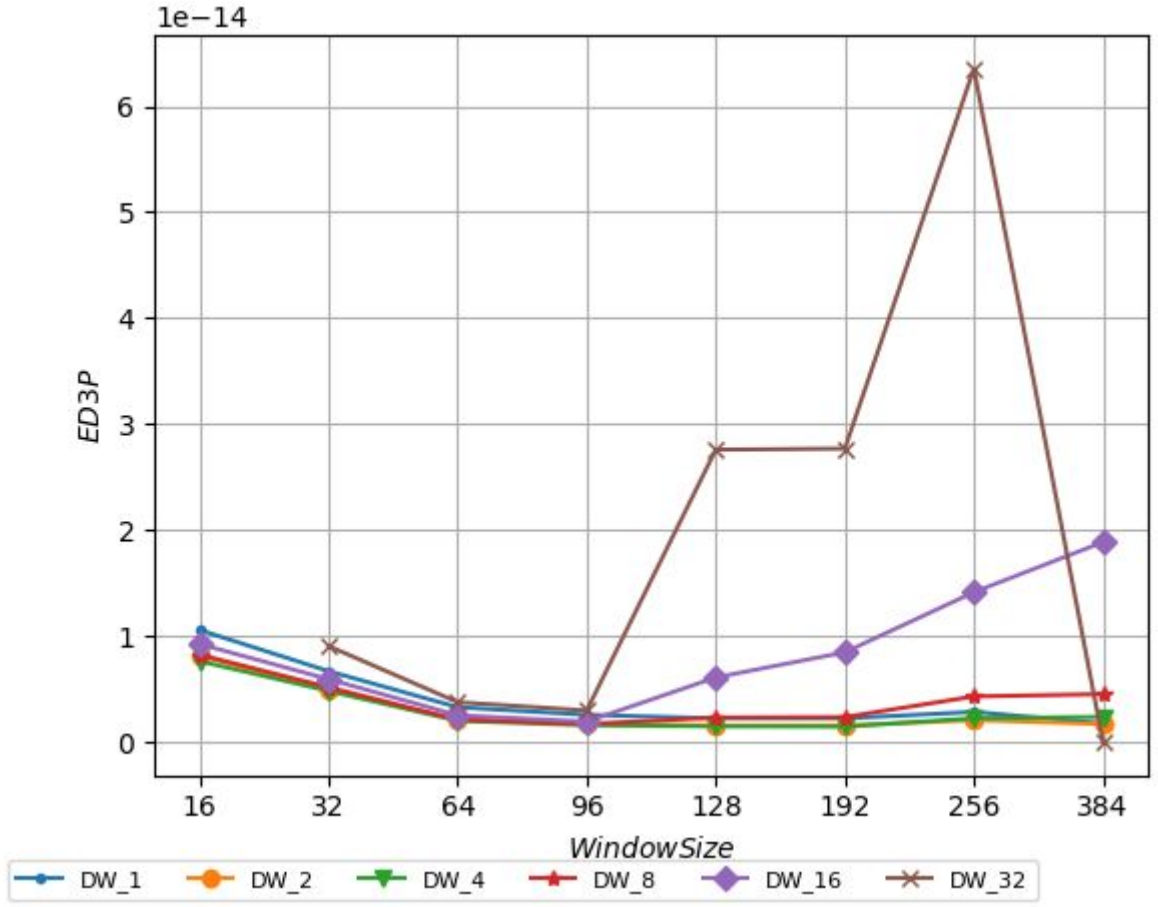
**sjeng**


0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
ED3P
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

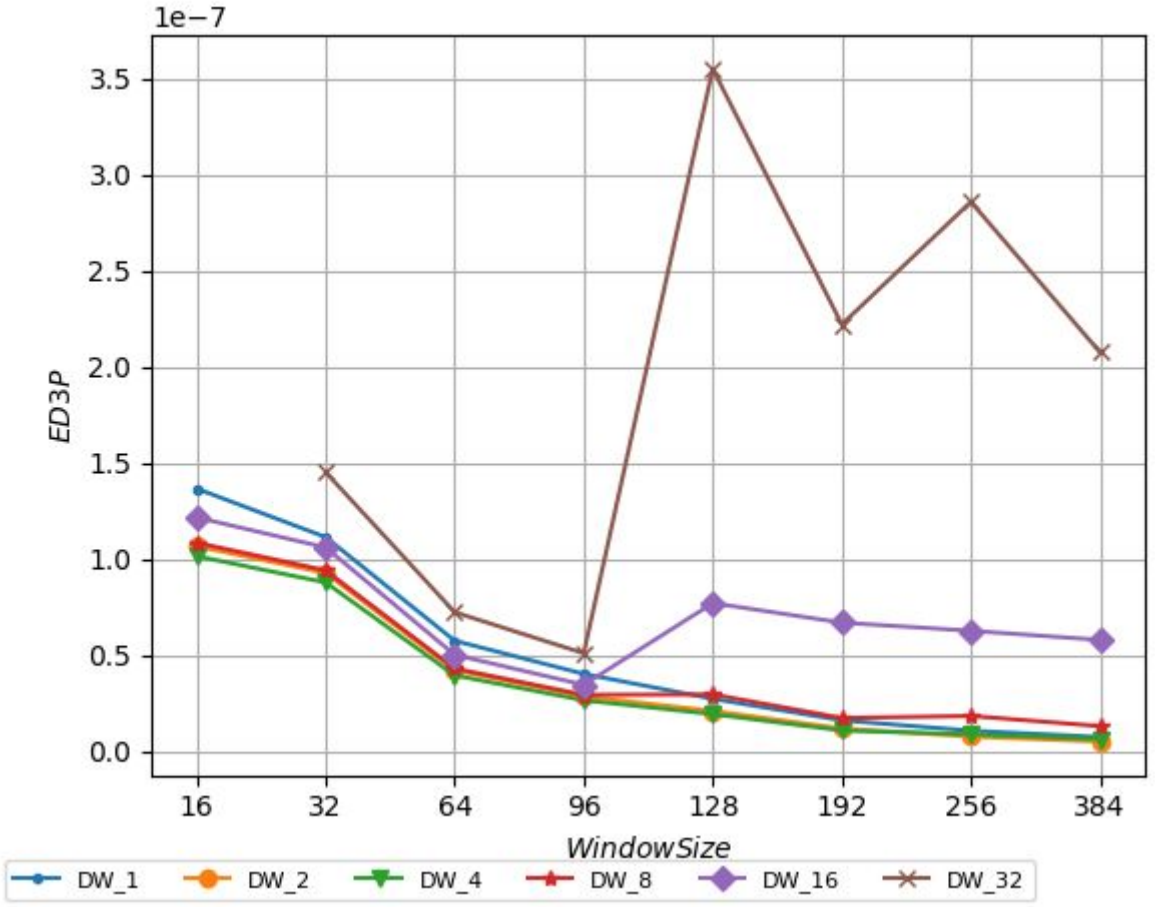
**GemsFDTD**


ED3P
25
20
15
10
5
0
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

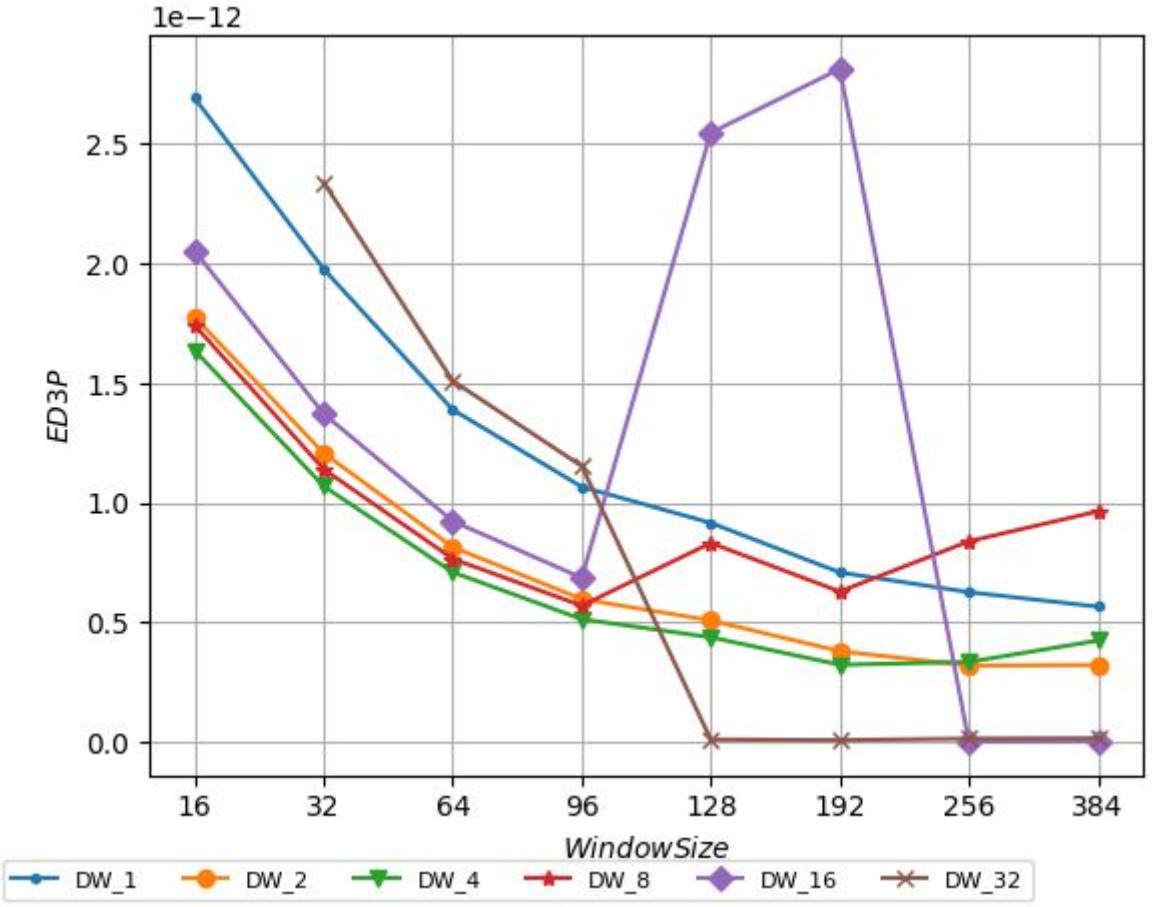
**omnetpp**


1e−14
6
5
4
3
2
1
0
ED3P
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**astar**

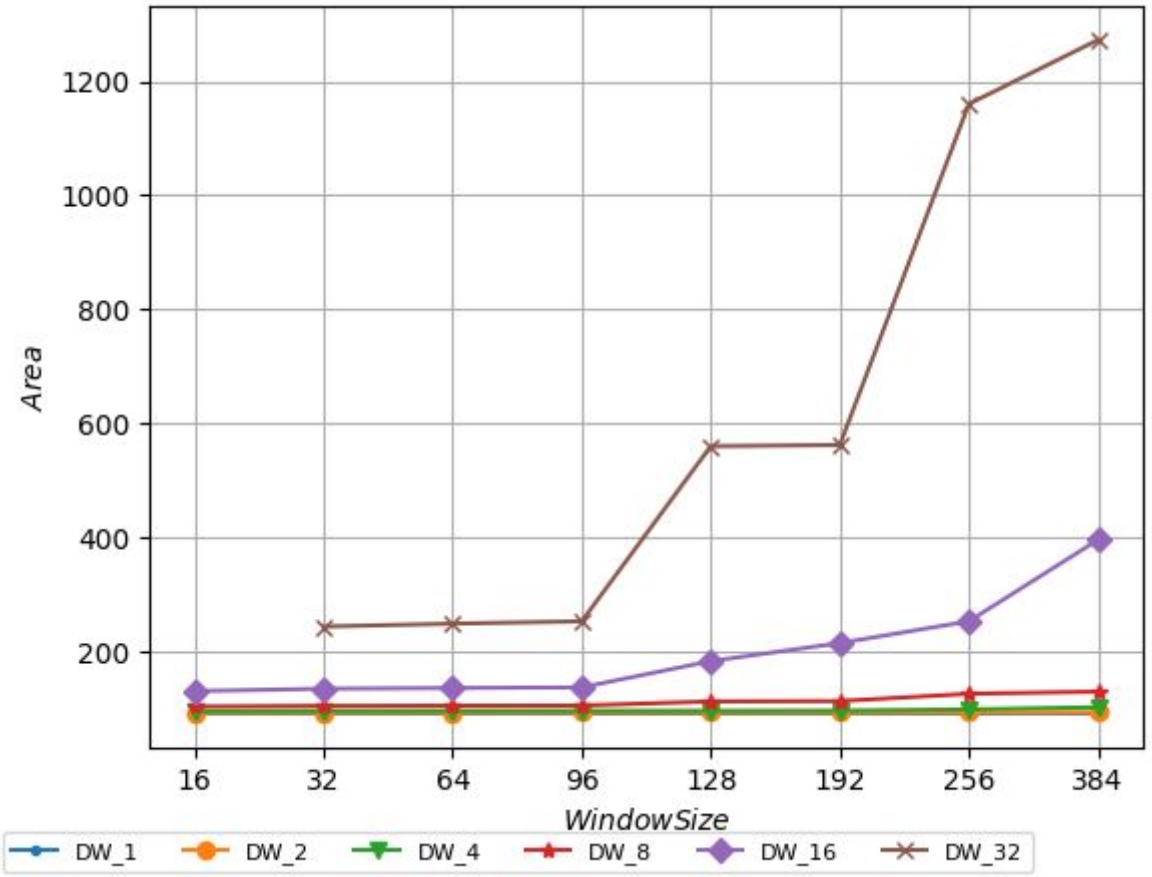

1e−7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
ED3P
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**xalancbmk**

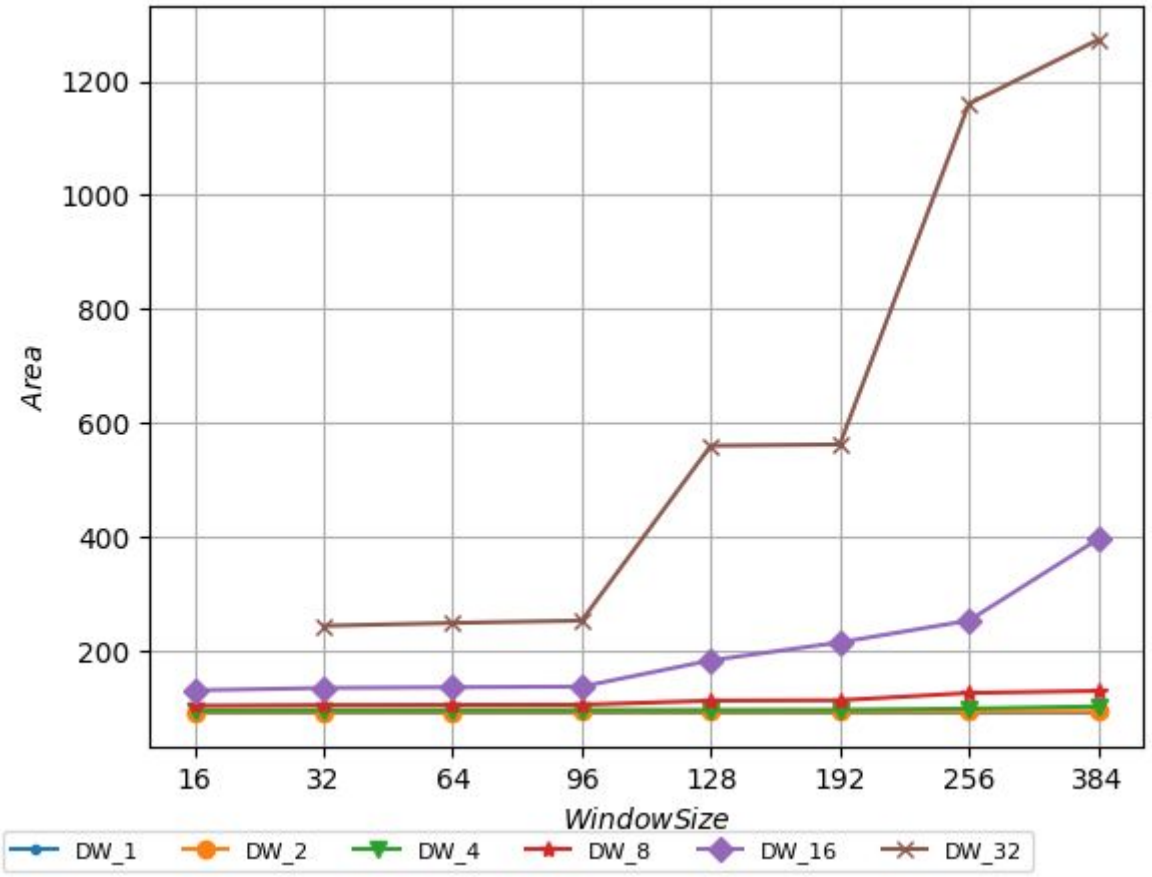

1e−12
2.5
2.0
1.5
1.0
0.5
0.0
ED3P
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**Για την Area(mm^2):**

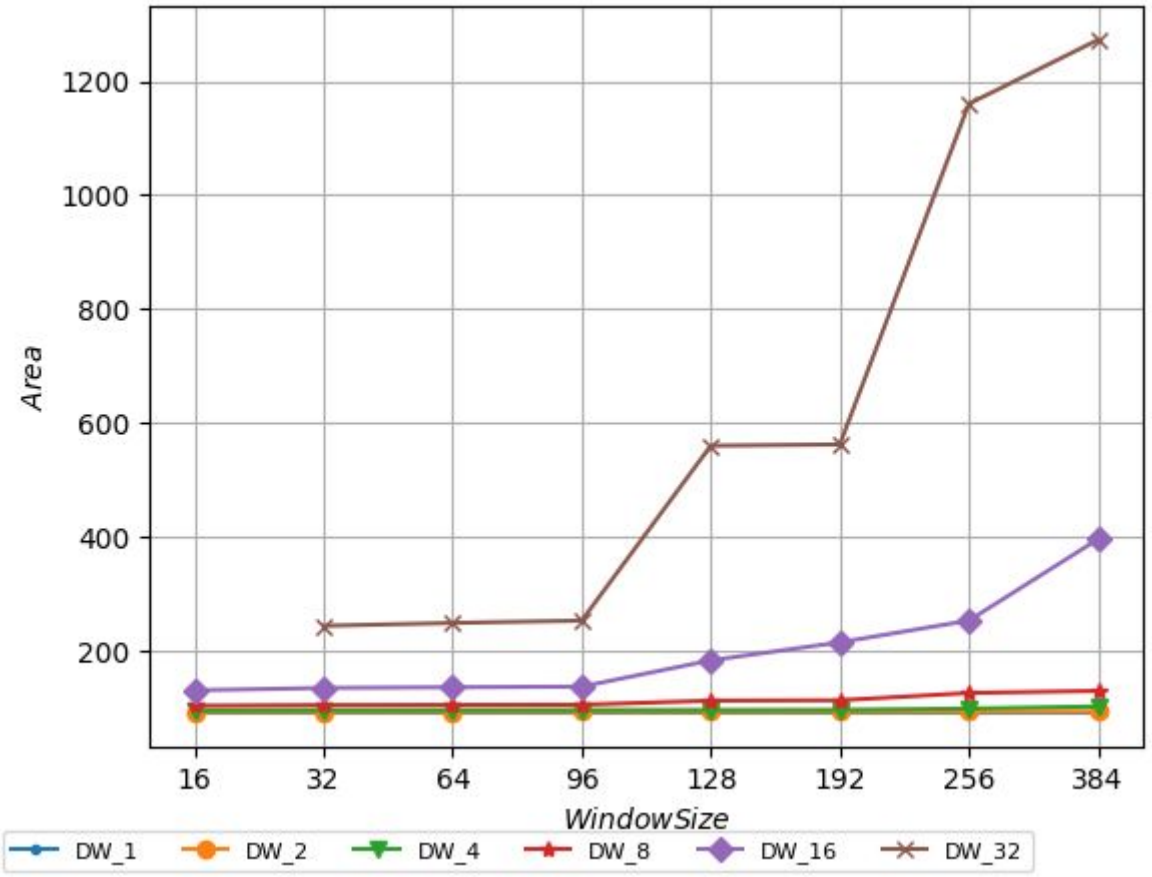
**gcc**



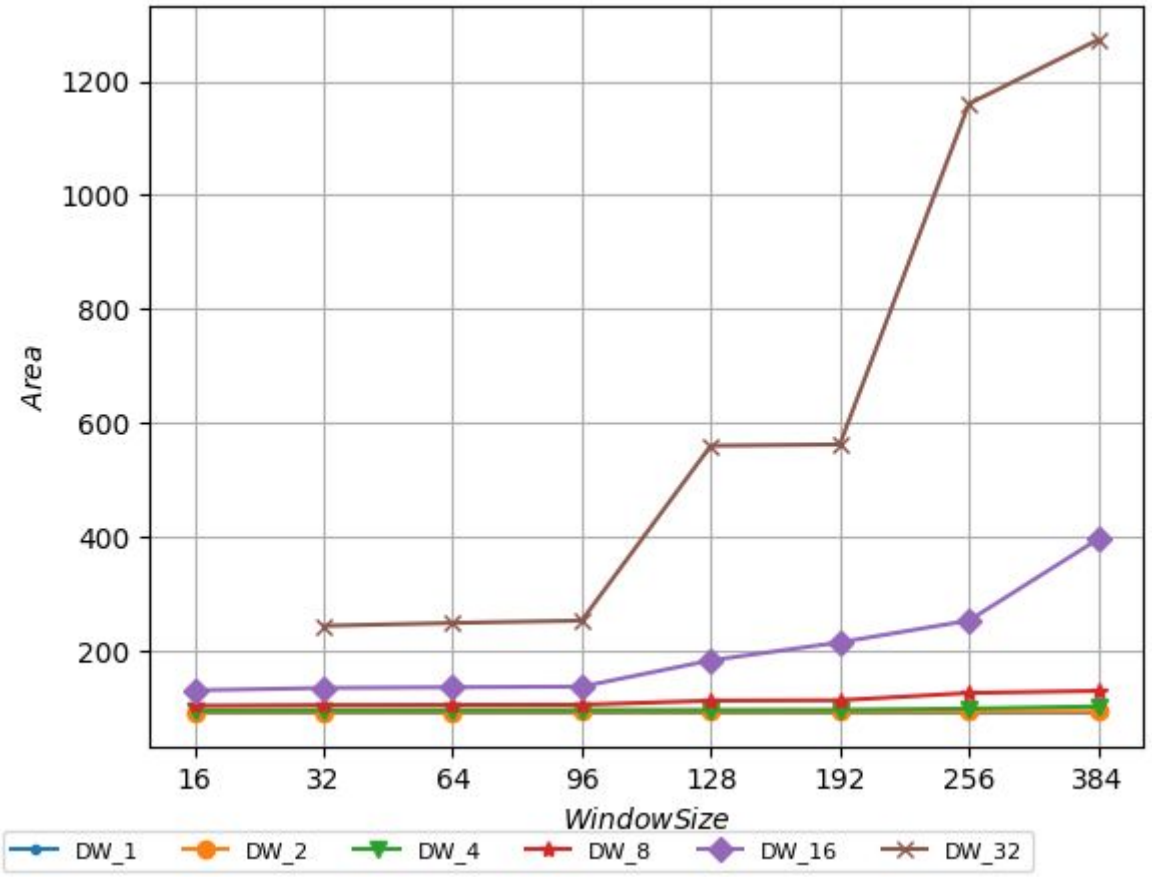
**mcf**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

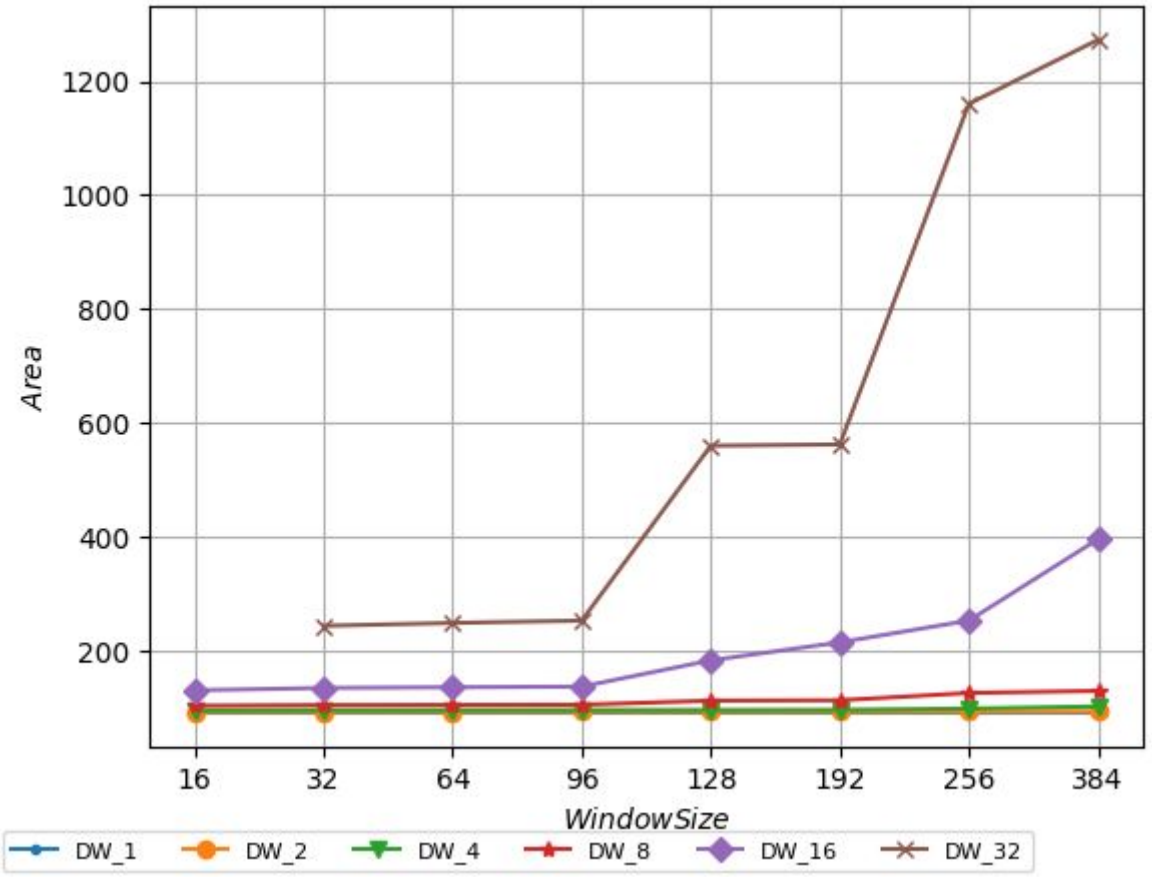
**zeusmp**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

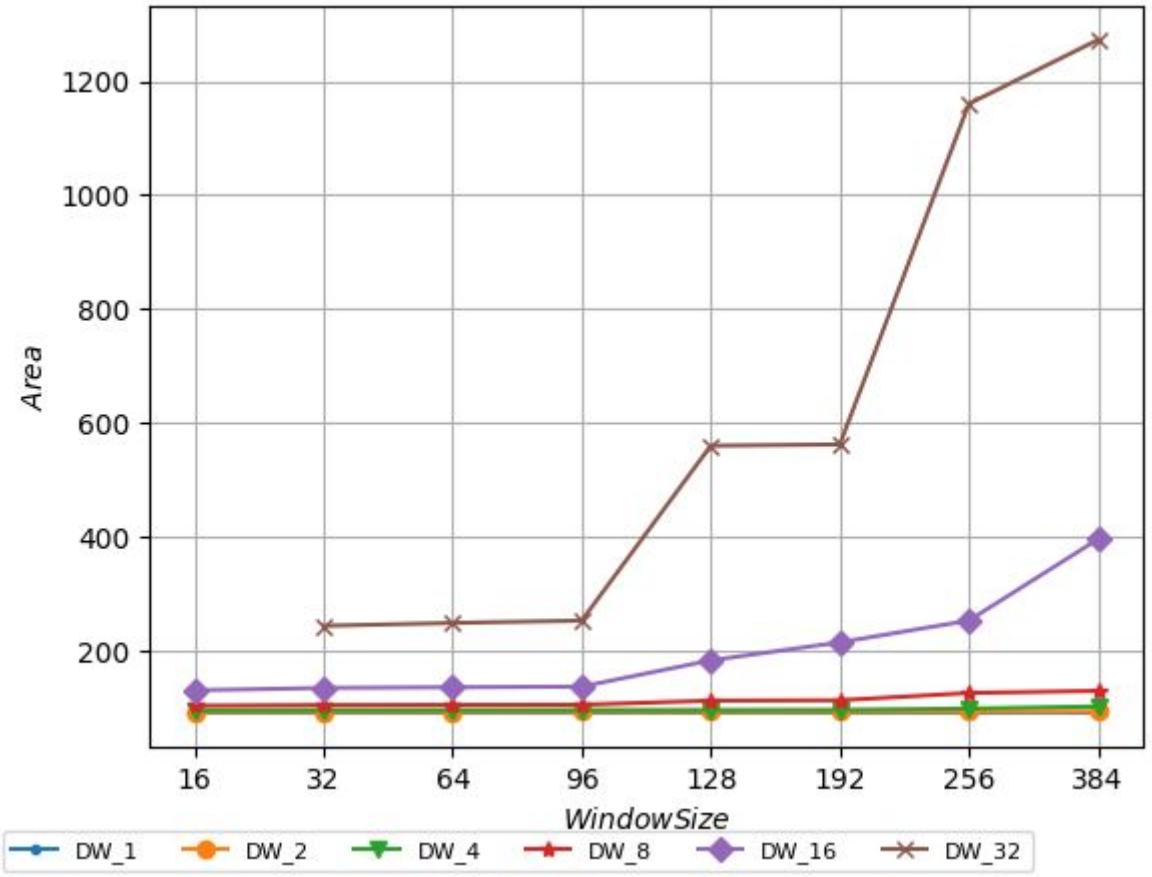
**cactusADM**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

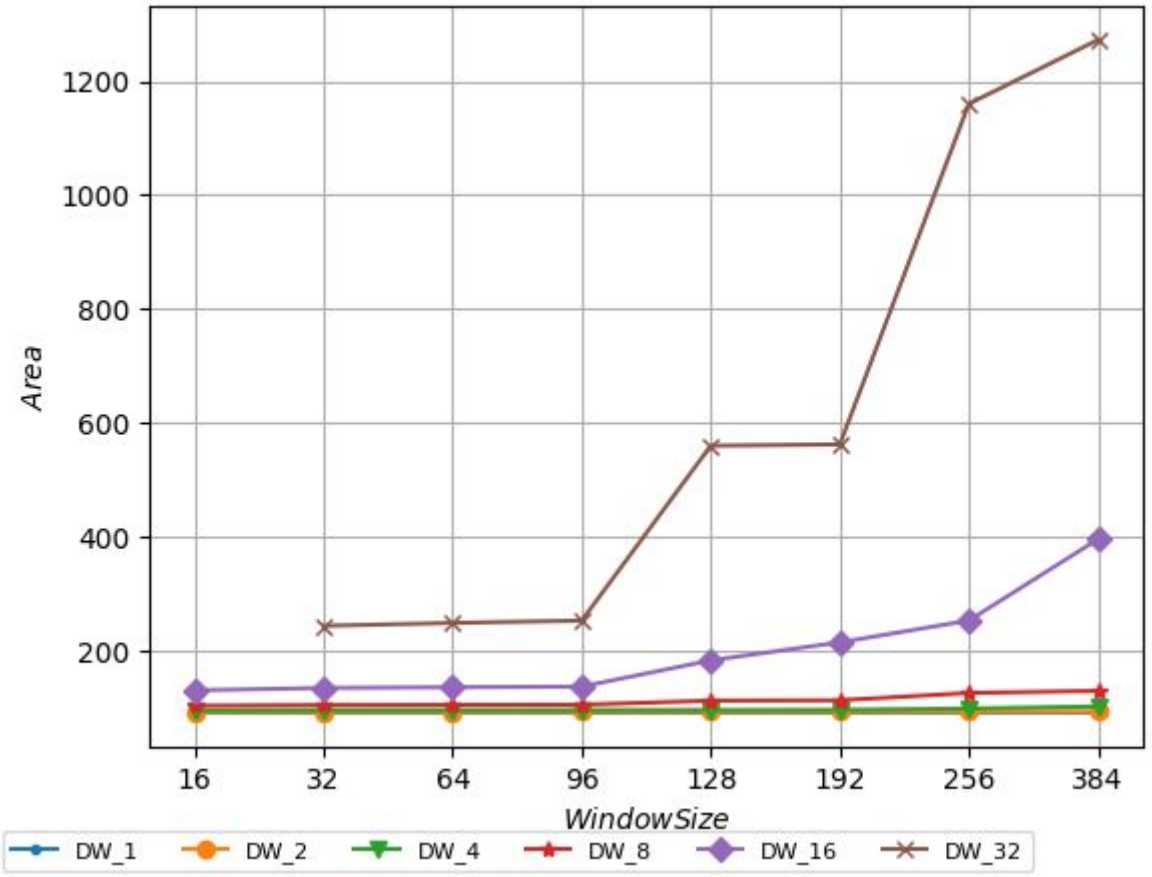
**gobmk**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

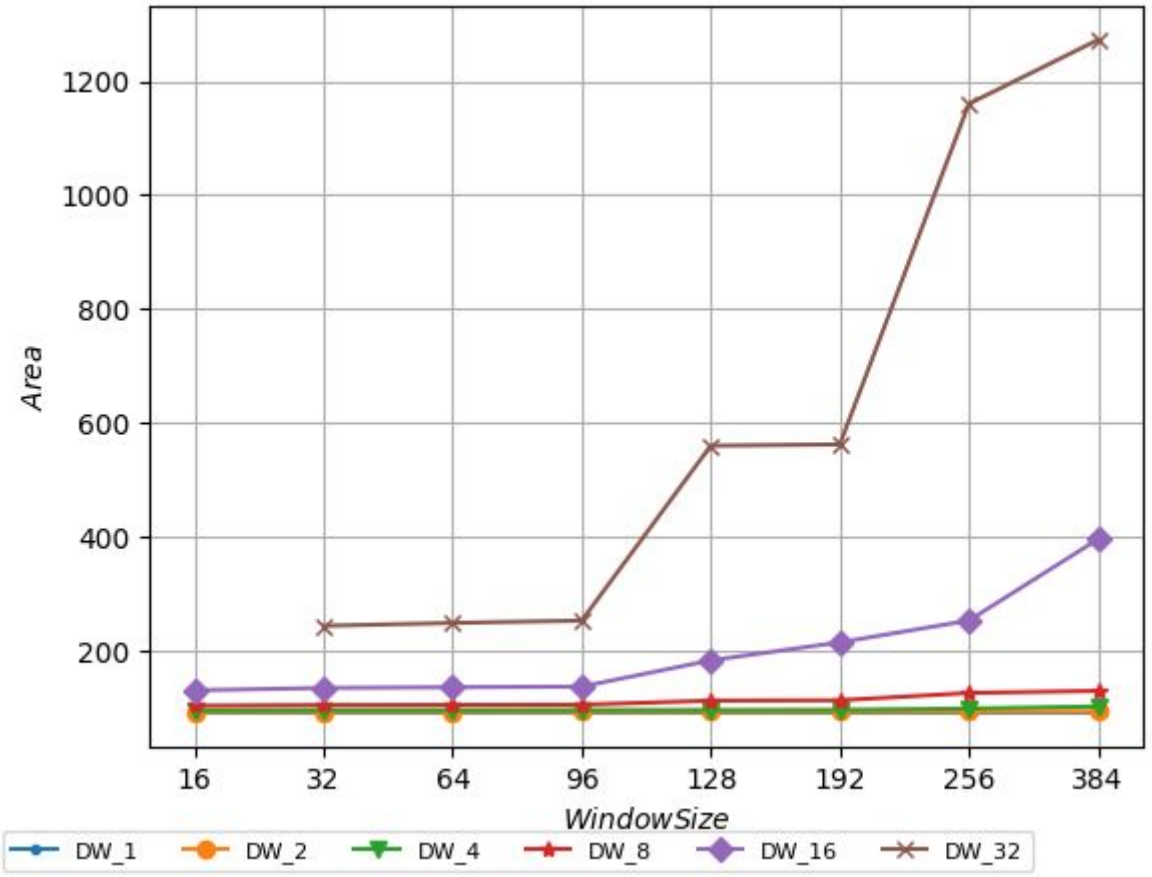
**soplex**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

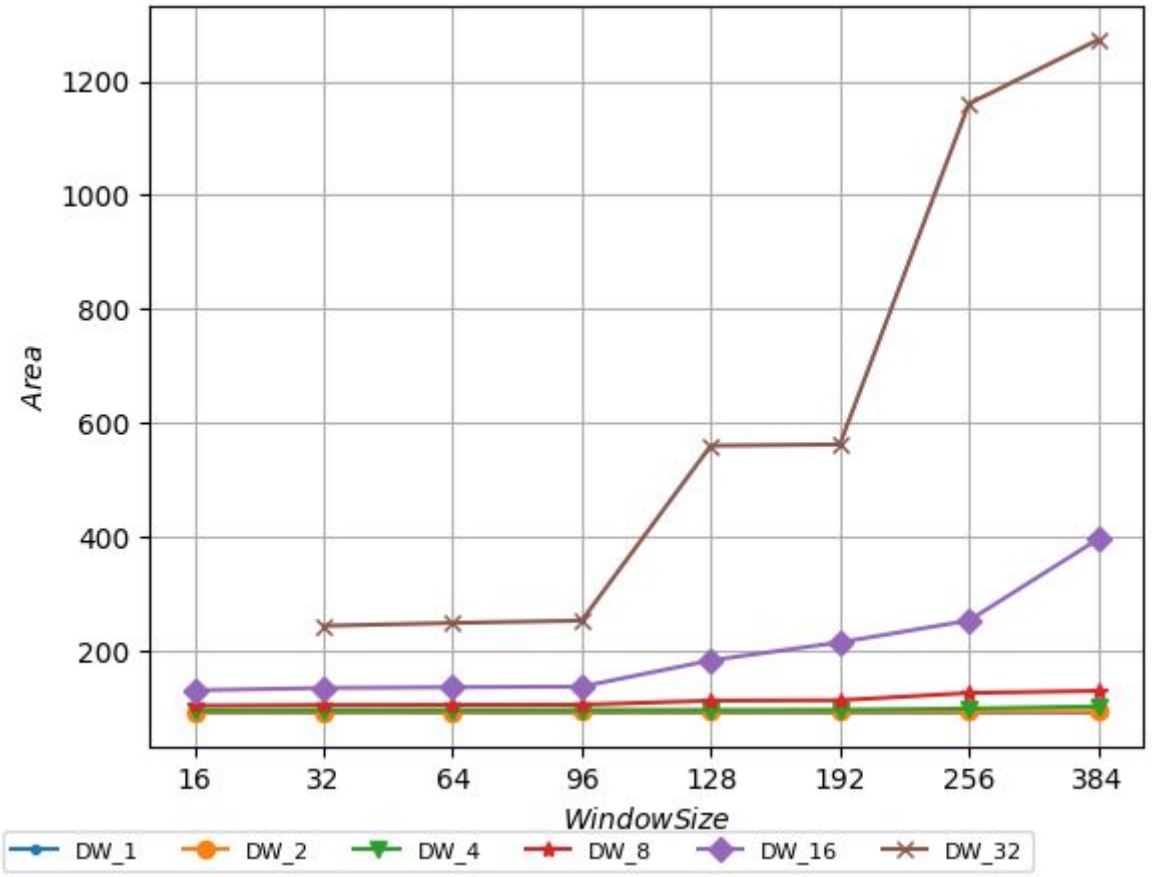
**hmmer**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

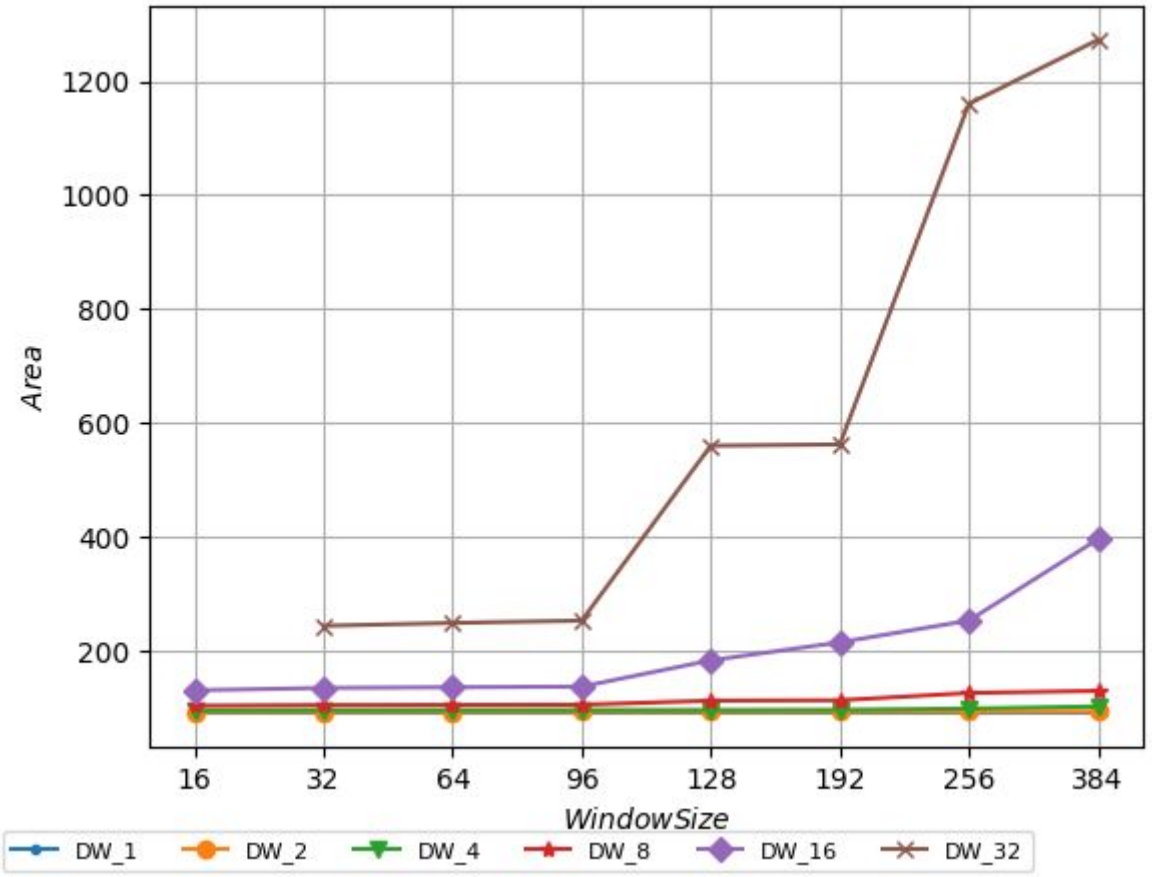
**sjeng**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

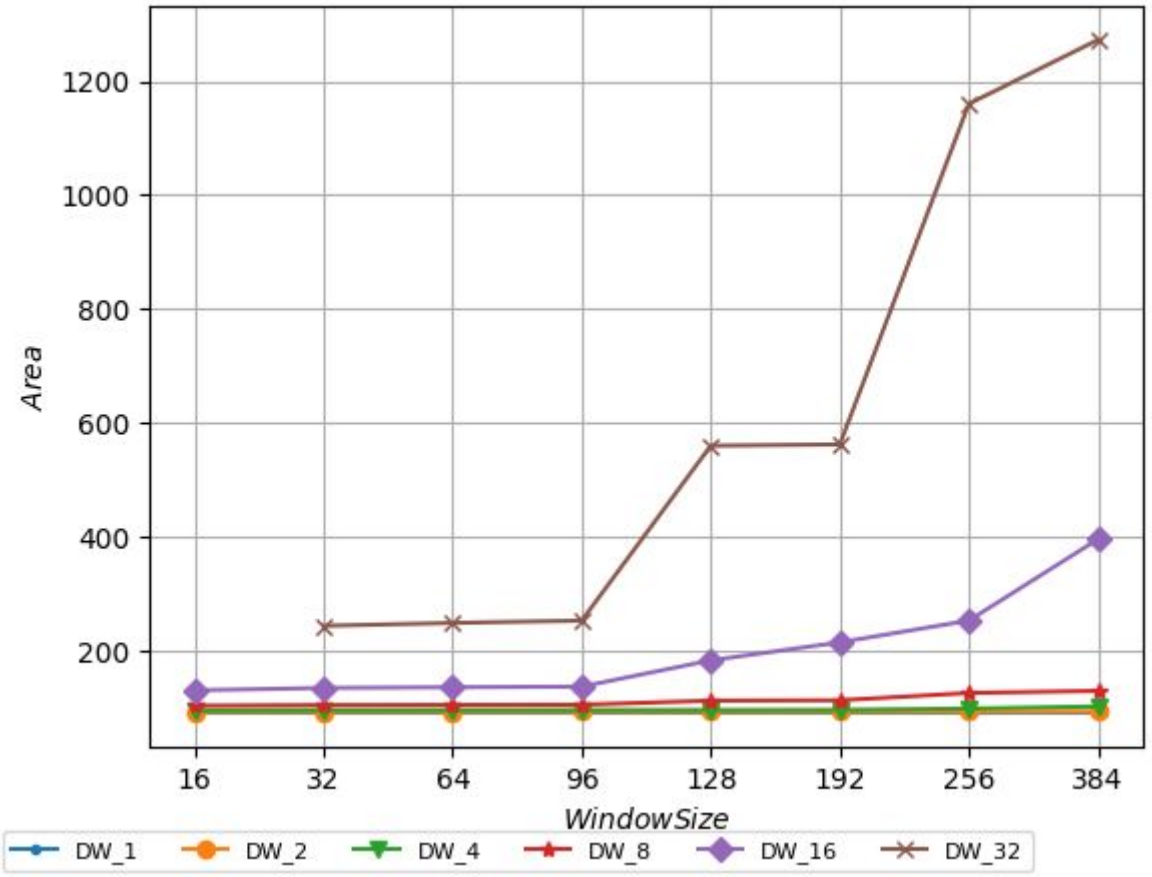
**GemsFDTD**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

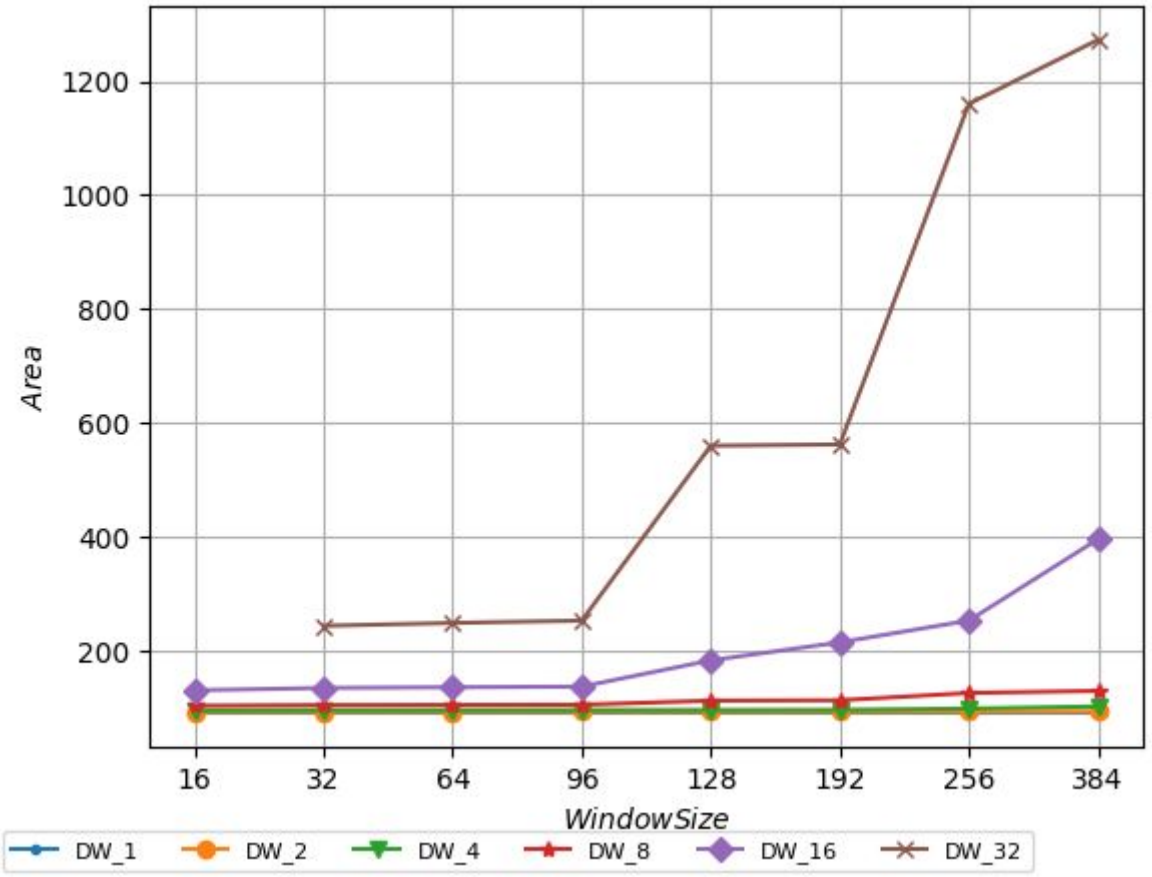
**omnetpp**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**astar**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**xalancbmk**


1200
1000
800
600
400
200
Area
16
32
64
96
128
192
256
384
WindowSize
DW_1
DW_2
DW_4
DW_8
DW_16
DW_32

**Συμπεράσματα για κατανάλωση ενέργεια και μέγεθος chip**

**ΕΡΩΤΗΣΗ (iii) Πώς επηρεάζει η κάθε παράμετρος την κατανάλωση ενέργειας και το μέγεθος του τσιπ;**

Γενικός Κανόνας - Συμπέρασμα: Η αύξηση dispatch_width και window_size οδηγεί σε αύξηση του μεγέθους του chip και της κατανάλωσης ενέργειας

- Για μικρές τιμές του dispatch width, δηλαδη έως και 8, το window size δεν παίζει τόσο μεγάλο ρόλο ούτε στην αύξηση της ενεργειακής δαπάνης, ούτε στο μέγεθος του τσιπ.
- Για dispatch width 16 και 32, η αύξηση του window size επηρεάζει και την ενέργεια που καταναλώνεται (όπως δείχνουν και τα διαγράμματα EDP, ED^2P, ED^3P) και το μέγεθος του chip (όπως φαίνεται στο τελευταίο διάγραμμα).
- Ειδικά για dispatch width ίσο με 32 και μέγεθος παραθύρου ίσο με 256 και 384 η διαφορα είναι πολύ μεγάλη χωρίς την αντίστοιχη αύξηση της επίδοσης. Η παραπάνω όμως αύξηση συμβαίνει μόνο στο core, καθώς το μέγεθος και η κατανάλωση της cache μένει σταθερή για όλα τα χαρακτηριστικά.
- Παρατηρούμε ότι για μεγάλα window size και dispatch width έχουμε τεράστια αύξηση σε μέγεθος τσιπ και κατανάλωση ενέργειας ενώ παράλληλα δεν έχουμε ανάλογη αύξηση της απόδοσης.
- Για window_size = 128 και 192 παρατηρούμε ότι τόσο στην κατανάλωση όσο και στο μέγεθος του τσιπ, δεν παρατηρείται κάποια ιδιαίτερη αλλαγή.

Παρατήρηση: Τα συμπεράσματα που μπορεί να εξαγάγει κανείς από τις γραφικές των benchmark των ED^2Pκαι ED^3P, είναι ίδια με του EDP κατα σχήμα γραφικής αλλά διαφορετικά κατα έναν πολλαπλασιαστικό παράγοντα. Τοποθετούνται ωστόσο για πληρότητα.

**ΕΡΩΤΗΣΗ: (iv) Βρείτε τα αντίστοιχα χαρακτηριστικά (dispatch_width, window_size) για τον επεξεργαστή του προσωπικού σας υπολογιστή ή για κάποιον από τους σύγχρονους επεξεργαστές (π.χ. Broadwell, Skylake, Kabylake). Δικαιολογούνται οι τιμές που επιλέξανε σε αυτά τα συστήματα οι αρχιτέκτονες με βάση τις προσομοιώσεις που εκτελέσατε και τα συμπεράσματα στα οποία καταλήξατε; Θα είχε νόημα να ήταν διαφορετικές (π.χ. μεγαλύτερο window_size); Για ποιό λόγο πιστεύετε δεν κάνανε κάποια άλλη επιλογή;**

Ο επεξεργαστής του προσωπικού μου υπολογιστή είναι ο Intel Core TM i5-2435M και ανήκει στην αρχιτεκτονική Sandy Bridge. Έχει window size 128 και dispatch width 4. Οι τιμές αυτές συμπίπτουν με τις τιμές που βρήκαμε πειραματικά, όπου έχουμε πολύ μεγάλη αύξηση της απόδοσης με σχετικα μικρό κόστος. Είναι λογικό να έχει γίνει μια τέτοια επιλογή, αφού αν είχε μεγαλύτερο κόστος δε θα ήταν τοσο προσιτοί προς το ευρύ κοινό ενώ παράλληλα η απόδοση δε θα αυξανόταν μεγάλο βαθμο και θα αυξανόταν το ενεργειακό foorprint και θα οδηγούσε στην ανάγκη ύπαρξης πιο δυνατής μπαταρίας πιυ αυξάνει αρκετά το κόστος και επιδρά αρνητικά στο βάρος του laptop.

**Παράρτημα:**

**1.Κώδικας για εμφάνηση plot ipc και για τους 72 επεξεργαστες**

```
#!/usr/bin/env python

import sys, os
import itertools, operator
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np

def get_params_from_basename(basename):
    tokens = basename.split('.')
    bench = tokens[0]
    input_size = 'ref'
    dw = int(tokens[1].split('-')[0].split('_')[1])
    ws = int(tokens[1].split('-')[1].split('_')[1])
    return (bench, input_size, dw, ws)

def get_ipc_from_output_file(output_file):
    ipc = -999
    fp = open(output_file, "r")
    line = fp.readline()
    while line:
            if "IPC" in line:
                    ipc = float(line.split()[2])
            line = fp.readline()

    fp.close()
    return ipc

def tuples_by_dispatch_width(tuples):
    ret = []
    tuples_sorted = sorted(tuples, key=operator.itemgetter(0))
    for key,group in itertools.groupby(tuples_sorted,operator.itemgetter(0)):
            ret.append((key, zip(*map(lambda x: x[1:], list(group)))))
    return ret

global_ws = [1,2,4,8,16,32,64,96,128,192,256,384]

if len(sys.argv) < 2:
    print "usage:", sys.argv[0], "<output_directories>"
    sys.exit(1)
```

```
results_tuples = []

for dirname in sys.argv[1:]:
    if dirname.endswith("/"):
            dirname = dirname[0:-1]
    basename = os.path.basename(dirname)
    output_file = dirname + "/sim.out"

    (bench, input_size, dispatch_width, window_size) =
get_params_from_basename(basename)
    ipc = get_ipc_from_output_file(output_file)
    results_tuples.append((dispatch_width, window_size, ipc))



markers = ['.', 'o', 'v', '*', 'D']
fig = plt.figure()
plt.grid(True)
ax = plt.subplot(111)
ax.set_xlabel("$Window Size$")
ax.set_ylabel("$IPC$")

i = 0
tuples_by_dw = tuples_by_dispatch_width(results_tuples)
for tuple in tuples_by_dw:
    dw = tuple[0]
    ws_axis = tuple[1][0]
    ipc_axis = tuple[1][1]
    x_ticks = np.arange(0, len(global_ws))
    x_labels = map(str, global_ws)
    ax.xaxis.set_ticks(x_ticks)
    ax.xaxis.set_ticklabels(x_labels)

    print x_ticks
    print ipc_axis
    ax.plot(x_ticks, ipc_axis, label="DW_"+str(dw), marker=markers[i%len(markers)])
    i = i + 1

lgd = ax.legend(ncol=len(tuples_by_dw), bbox_to_anchor=(0.9, -0.1), prop={'size':8})
plt.savefig(bench+'-'+input_size+'.ipc.png', bbox_extra_artists=(lgd,), bbox_inches='tight')
```

**2.Κώδικας για εμφάνιση plot ipc και για τους 57 επεξεργαστες**

```
#!/usr/bin/env python

import sys, os
import itertools, operator
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np

def get_params_from_basename(basename):
    tokens = basename.split('.')
    bench = tokens[0]
    print(tokens)
    input_size = 'ref'
    dw = int(tokens[1].split('-')[0].split('_')[1])
    ws = int(tokens[1].split('-')[1].split('_')[1])
    return (bench, input_size, dw, ws)

def get_ipc_from_output_file(output_file):
    ipc = -999
    fp = open(output_file, "r")
    line = fp.readline()
    while line:
            if "IPC" in line:
                    ipc = float(line.split()[2])
            line = fp.readline()

    fp.close()
    return ipc

def tuples_by_dispatch_width(tuples):
    ret = []
    tuples_sorted = sorted(tuples, key=operator.itemgetter(0))
    for key,group in itertools.groupby(tuples_sorted,operator.itemgetter(0)):
            ret.append((key, list(zip(*map(lambda x: x[1:], list(group))))))
    return ret

global_ws = [1,2,4,8,16,32,64,96,128,192,256,384]

if len(sys.argv) < 2:
    print ("usage:", sys.argv[0], "<output_directories>")
    sys.exit(1)

results_tuples = []
```

```
for dirname in sys.argv[1:]:
    if dirname.endswith("/"):
            dirname = dirname[0:-1]
    basename = os.path.basename(dirname)
    output_file = dirname + "/sim.out"

    (bench, input_size, dispatch_width, window_size) =
get_params_from_basename(basename)
    ipc = get_ipc_from_output_file(output_file)
    results_tuples.append((dispatch_width, window_size, ipc))


markers = ['.', 'o', 'v', '*', 'D', 'x']
fig = plt.figure()
plt.grid(True)
ax = plt.subplot(111)
ax.set_xlabel("$Window Size$")
ax.set_ylabel("$IPC$")

i = 0
tuples_by_dw = tuples_by_dispatch_width(results_tuples)
print(tuples_by_dw)
for tuple in tuples_by_dw:
    dw = tuple[0]
    ws_axis = tuple[1][0]
    ipc_axis = tuple[1][1]
    x_ticks = np.arange(i, len(global_ws))
    x_labels = map(str, global_ws[i:len(global_ws)])
    if (i==0):
            ax.xaxis.set_ticks(x_ticks)
            ax.xaxis.set_ticklabels(x_labels)

    print (x_ticks)
    print (ipc_axis)
    ax.plot(x_ticks, ipc_axis, label="DW_"+str(dw), marker=markers[i%len(markers)])
    i = i + 1

lgd = ax.legend(ncol=len(tuples_by_dw), bbox_to_anchor=(0.9, -0.1), prop={'size':8})
plt.savefig(bench+'-'+input_size+'.ipc.png', bbox_extra_artists=(lgd,), bbox_inches='tight')
```

**3.Κώδικας για εμφάνιση plot edp**

```
#!/usr/bin/env python

import sys, os
import itertools, operator
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np

def get_params_from_basename(basename):
    tokens = basename.split('.')
    bench = tokens[0]
    input_size = 'ref'
    dw = int(tokens[1].split('-')[0].split('_')[1])
    ws = int(tokens[1].split('-')[1].split('_')[1])
    return (bench, input_size, dw, ws)

def get_edp_from_output_file(output_file1, output_file2):
    edp = -999
    fp1 = open(output_file1, "r")
    line = fp1.readline()
    while line:
        if "IPC" in line:
            line = fp1.readline()
            runtime = float(line.split()[3]) / float(10**9)
        line = fp1.readline()
    fp1.close()

    fp2 = open(output_file2, "r")
    line = fp2.readline()
    while line:
        if "total" in line:
            energy = float(line.split()[3])
        line = fp2.readline()
    fp2.close()

    edp = energy * runtime

    return edp

def tuples_by_dispatch_width(tuples):
    ret = []
    tuples_sorted = sorted(tuples, key=operator.itemgetter(0))
    for key,group in itertools.groupby(tuples_sorted,operator.itemgetter(0)):
```

```
            ret.append((key, list(zip(*map(lambda x: x[1:], list(group))))))
    return ret

global_ws = [16,32,64,96,128,192,256,384]

if len(sys.argv) < 2:
    print ("usage:", sys.argv[0], "<output_directories>")
    sys.exit(1)

results_tuples = []

for dirname in sys.argv[1:]:
    if dirname.endswith("/"):
            dirname = dirname[0:-1]
    basename = os.path.basename(dirname)
    output_file1 = dirname + "/sim.out"
    output_file2 = dirname + "/power.total.out"

    (bench, input_size, dispatch_width, window_size) =
get_params_from_basename(basename)
    edp = get_edp_from_output_file(output_file1, output_file2)
    results_tuples.append((dispatch_width, window_size, edp))


markers = ['.', 'o', 'v', '*', 'D', 'x']
fig = plt.figure()
plt.grid(True)
ax = plt.subplot(111)
ax.set_xlabel("$Window Size$")
ax.set_ylabel("$EDP$")

i = 0
tuples_by_dw = tuples_by_dispatch_width(results_tuples)
for tuple in tuples_by_dw:
    dw = tuple[0]
    ws_axis = tuple[1][0]
    edp_axis = tuple[1][1]
    if (i==5):
            x_ticks = np.arange(1, len(global_ws))
            x_labels = map(str, global_ws[1:len(global_ws)])
    else:
            x_ticks = np.arange(0, len(global_ws))
            x_labels = map(str, global_ws)
    if (i==0):
            ax.xaxis.set_ticks(x_ticks)
            ax.xaxis.set_ticklabels(x_labels)
```

```
    print (x_ticks)
    print (edp_axis)
    ax.plot(x_ticks, edp_axis, label="DW_"+str(dw), marker=markers[i%len(markers)])
    i = i + 1

lgd = ax.legend(ncol=len(tuples_by_dw), bbox_to_anchor=(0.9, -0.1), prop={'size':8})
plt.savefig(bench+'-'+input_size+'.edp.png', bbox_extra_artists=(lgd,), bbox_inches='tight')
```

**4.Κώδικας για εμφάνιση plot ed^2p**

```
#!/usr/bin/env python

import sys, os
import itertools, operator
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np

def get_params_from_basename(basename):
    tokens = basename.split('.')
    bench = tokens[0]
    input_size = 'ref'
    dw = int(tokens[1].split('-')[0].split('_')[1])
    ws = int(tokens[1].split('-')[1].split('_')[1])
    return (bench, input_size, dw, ws)

def get_edp_from_output_file(output_file1, output_file2):
    edp = -999
    fp1 = open(output_file1, "r")
    line = fp1.readline()
    while line:
            if "IPC" in line:
                    line = fp1.readline()
                    runtime = float(line.split()[3]) / float(10**9)
            line = fp1.readline()
    fp1.close()

    fp2 = open(output_file2, "r")
    line = fp2.readline()
    while line:
            if "total" in line:
                    energy = float(line.split()[3])
            line = fp2.readline()
    fp2.close()
```

```
        #calculate edp2
    edp = energy * runtime * runtime

    return edp

def tuples_by_dispatch_width(tuples):
    ret = []
    tuples_sorted = sorted(tuples, key=operator.itemgetter(0))
    for key,group in itertools.groupby(tuples_sorted,operator.itemgetter(0)):
        ret.append((key, list(zip(*map(lambda x: x[1:], list(group))))))
    return ret

global_ws = [16,32,64,96,128,192,256,384]

if len(sys.argv) < 2:
    print ("usage:", sys.argv[0], "<output_directories>")
    sys.exit(1)

results_tuples = []

for dirname in sys.argv[1:]:
    if dirname.endswith("/"):
        dirname = dirname[0:-1]
    basename = os.path.basename(dirname)
    output_file1 = dirname + "/sim.out"
    output_file2 = dirname + "/power.total.out"

    (bench, input_size, dispatch_width, window_size) =
get_params_from_basename(basename)
    edp = get_edp_from_output_file(output_file1, output_file2)
    results_tuples.append((dispatch_width, window_size, edp))


markers = ['.', 'o', 'v', '*', 'D', 'x']
fig = plt.figure()
plt.grid(True)
ax = plt.subplot(111)
ax.set_xlabel("$Window Size$")
ax.set_ylabel("$ED^{2}P$")

i = 0
tuples_by_dw = tuples_by_dispatch_width(results_tuples)
for tuple in tuples_by_dw:
    dw = tuple[0]
    ws_axis = tuple[1][0]
    edp_axis = tuple[1][1]
```

```
    if (i==5):
        x_ticks = np.arange(1, len(global_ws))
        x_labels = map(str, global_ws[1:len(global_ws)])
    else:
        x_ticks = np.arange(0, len(global_ws))
        x_labels = map(str, global_ws)
    if (i==0):
        ax.xaxis.set_ticks(x_ticks)
        ax.xaxis.set_ticklabels(x_labels)

    print (x_ticks)
    print (edp_axis)
    ax.plot(x_ticks, edp_axis, label="DW_"+str(dw), marker=markers[i%len(markers)])
    i = i + 1

lgd = ax.legend(ncol=len(tuples_by_dw), bbox_to_anchor=(0.9, -0.1), prop={'size':8})
plt.savefig(bench+'-'+input_size+'.edp2.png', bbox_extra_artists=(lgd,), bbox_inches='tight')
```

**5.Κώδικας για εμφάνιση plot ed^3p**

```
#!/usr/bin/env python

import sys, os
import itertools, operator
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np

def get_params_from_basename(basename):
    tokens = basename.split('.')
    bench = tokens[0]
    input_size = 'ref'
    dw = int(tokens[1].split('-')[0].split('_')[1])
    ws = int(tokens[1].split('-')[1].split('_')[1])
    return (bench, input_size, dw, ws)

def get_edp_from_output_file(output_file1, output_file2):
    edp = -999
    fp1 = open(output_file1, "r")
    line = fp1.readline()
    while line:
        if "IPC" in line:
            line = fp1.readline()
            runtime = float(line.split()[3]) / float(10**9)
        line = fp1.readline()
```

```
    fp1.close()

    fp2 = open(output_file2, "r")
    line = fp2.readline()
    while line:
            if "total" in line:
                    energy = float(line.split()[3])
            line = fp2.readline()
    fp2.close()
           #calculate edp3
    edp = energy * runtime * runtime * runtime

    return edp

def tuples_by_dispatch_width(tuples):
    ret = []
    tuples_sorted = sorted(tuples, key=operator.itemgetter(0))
    for key,group in itertools.groupby(tuples_sorted,operator.itemgetter(0)):
            ret.append((key, list(zip(*map(lambda x: x[1:], list(group))))))
    return ret

global_ws = [16,32,64,96,128,192,256,384]

if len(sys.argv) < 2:
    print ("usage:", sys.argv[0], "<output_directories>")
    sys.exit(1)

results_tuples = []

for dirname in sys.argv[1:]:
    if dirname.endswith("/"):
            dirname = dirname[0:-1]
    basename = os.path.basename(dirname)
    output_file1 = dirname + "/sim.out"
    output_file2 = dirname + "/power.total.out"

    (bench, input_size, dispatch_width, window_size) =
get_params_from_basename(basename)
    edp = get_edp_from_output_file(output_file1, output_file2)
    results_tuples.append((dispatch_width, window_size, edp))



markers = ['.', 'o', 'v', '*', 'D', 'x']
fig = plt.figure()
plt.grid(True)
ax = plt.subplot(111)
```

```
ax.set_xlabel("$Window Size$")
ax.set_ylabel("$ED{3}P$")

i = 0
tuples_by_dw = tuples_by_dispatch_width(results_tuples)
for tuple in tuples_by_dw:
    dw = tuple[0]
    ws_axis = tuple[1][0]
    edp_axis = tuple[1][1]
    if (i==5):
            x_ticks = np.arange(1, len(global_ws))
            x_labels = map(str, global_ws[1:len(global_ws)])
    else:
            x_ticks = np.arange(0, len(global_ws))
            x_labels = map(str, global_ws)
    if (i==0):
            ax.xaxis.set_ticks(x_ticks)
            ax.xaxis.set_ticklabels(x_labels)

    print (x_ticks)
    print (edp_axis)
    ax.plot(x_ticks, edp_axis, label="DW_"+str(dw), marker=markers[i%len(markers)])
    i = i + 1

lgd = ax.legend(ncol=len(tuples_by_dw), bbox_to_anchor=(0.9, -0.1), prop={'size':8})
plt.savefig(bench+'-'+input_size+'.edp3.png', bbox_extra_artists=(lgd,), bbox_inches='tight')
```

**6.Κώδικας για εμφάνιση plot area**

```
#!/usr/bin/env python

import sys, os
import itertools, operator
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np

def get_params_from_basename(basename):
    tokens = basename.split('.')
    bench = tokens[0]
    print(tokens)
    input_size = 'ref'
    dw = int(tokens[1].split('-')[0].split('_')[1])
    ws = int(tokens[1].split('-')[1].split('_')[1])
    return (bench, input_size, dw, ws)
```

```
def get_area_from_output_file(output_file):
    area = -999
    fp = open(output_file, "r")
    line = fp.readline()
    while line:
            if "Processor:" in line:
                    line = fp.readline()
                    area = float(line.split()[2])
            line = fp.readline()

    fp.close()
    return area

def tuples_by_dispatch_width(tuples):
    ret = []
    tuples_sorted = sorted(tuples, key=operator.itemgetter(0))
    for key,group in itertools.groupby(tuples_sorted,operator.itemgetter(0)):
            ret.append((key, list(zip(*map(lambda x: x[1:], list(group))))))
    return ret

global_ws = [16,32,64,96,128,192,256,384]

if len(sys.argv) < 2:
    print ("usage:", sys.argv[0], "<output_directories>")
    sys.exit(1)

results_tuples = []

for dirname in sys.argv[1:]:
    if dirname.endswith("/"):
            dirname = dirname[0:-1]
    basename = os.path.basename(dirname)
    output_file = dirname + "/power.txt"

    (bench, input_size, dispatch_width, window_size) =
get_params_from_basename(basename)
    area = get_area_from_output_file(output_file)
    results_tuples.append((dispatch_width, window_size, area))


markers = ['.', 'o', 'v', '*', 'D', 'x']
fig = plt.figure()
plt.grid(True)
ax = plt.subplot(111)
ax.set_xlabel("$Window Size$")
```

```
ax.set_ylabel("$Area$")

i = 0
tuples_by_dw = tuples_by_dispatch_width(results_tuples)
print(tuples_by_dw)
for tuple in tuples_by_dw:
    dw = tuple[0]
    ws_axis = tuple[1][0]
    area_axis = tuple[1][1]
    if (i==5):
            x_ticks = np.arange(1, len(global_ws))
            x_labels = map(str, global_ws[1:len(global_ws)])
    else:
            x_ticks = np.arange(0, len(global_ws))
            x_labels = map(str, global_ws)
    if (i==0):
            ax.xaxis.set_ticks(x_ticks)
            ax.xaxis.set_ticklabels(x_labels)

    print (x_ticks)
    print (area_axis)
    ax.plot(x_ticks, area_axis, label="DW_"+str(dw), marker=markers[i%len(markers)])
    i = i + 1

lgd = ax.legend(ncol=len(tuples_by_dw), bbox_to_anchor=(0.9, -0.1), prop={'size':8})
plt.savefig(bench+'-'+input_size+'.area.png', bbox_extra_artists=(lgd,), bbox_inches='tight')
```