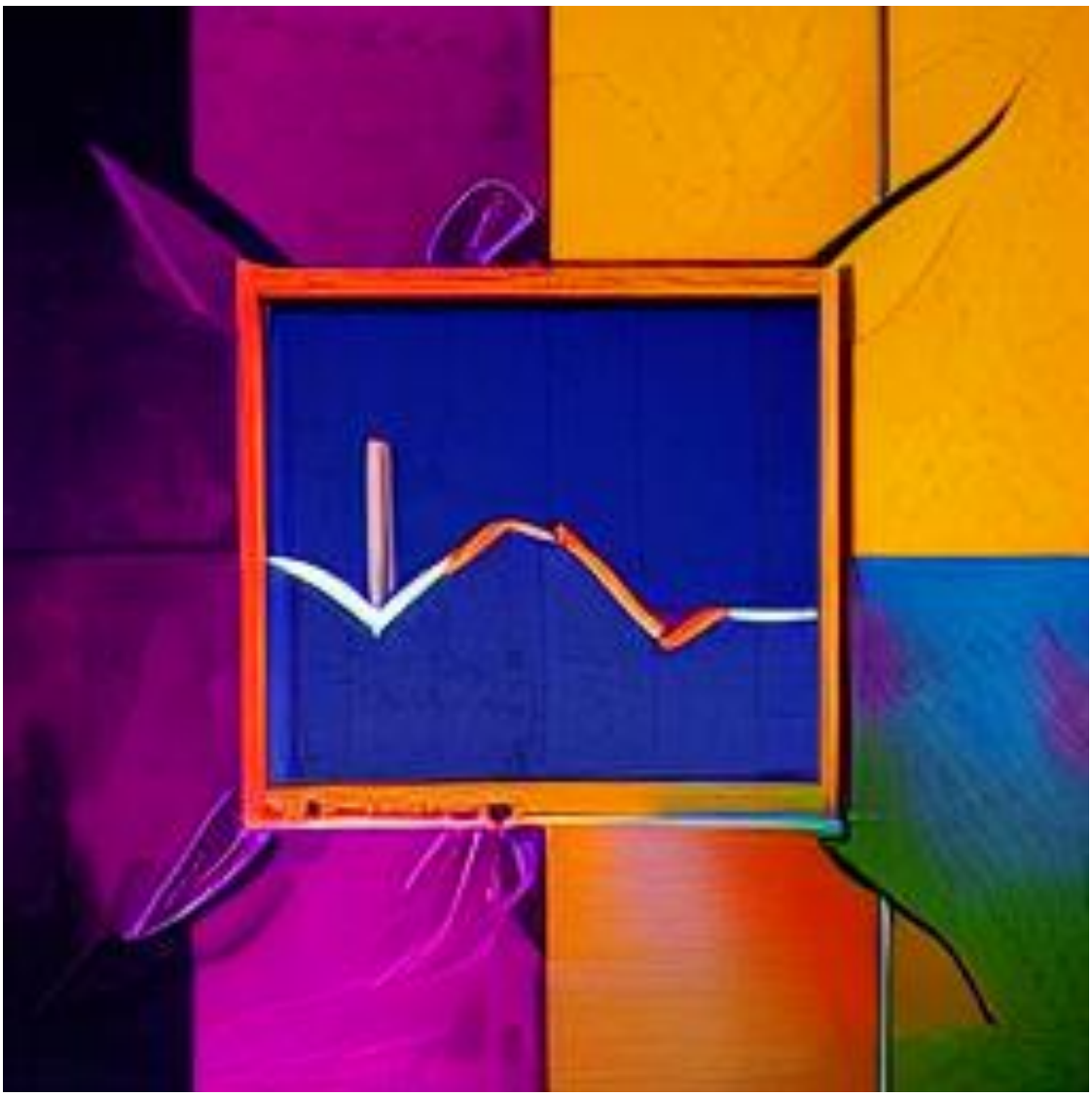
Εργασία Εξαμήνου

Γραμμική και συνδιαστική βελτιστοποίηση

Ιωάννης Τσάμπρας 4οετής φοιτητής ΗΜΤΥ /1066584/

‘Χρήση Mixed Integer Linear Programming (MILP) για την piece-wise μοντελοποίηση μη-γραμμικών συναρτήσεων κόστους μονάδων παραγωγής ενέργειας σε ELD προβλήματα’



Περιεχόμενα:

1. Το πρόβλημα του Economic Power Dispatch

Ο όρος Power Dispatch αναφέρεται σε ένα σύνολο αποφάσεων που λαμβάνονται σε συνεχή βάση (περίοδος ανανέωσης: ~30”-2’) και ρυθμίζουν παραμέτρους του ηλεκτρικού δικτύου με σκοπό την κάλυψη της μεταβαλλόμενης ζήτησης ισχύος από μια συλλογή έντονα διαφοροποιημένων μονάδων παραγωγής ενέργειας.

Στην κατηγορία προβλημάτων του Power Dispatch ανήκουν τα προβλήματα OPF όπου μέλημα είναι η ρύθμιση της ροής μιγαδικής ισχύος για ελαχιστοποίηση απωλειών μεταφοράς και υποδομών αντιστάθμισης, προβλήματα χρονικής απόκρισης μονάδων (ευσταθής έλεγχος) , μελέτες συσσωρευτών, γεφύρωσης δικτύων κ.α. Το υπο-πρόβλημα που πραγματεύεται αυτή η εργασία είναι η διαδικασία κατανομής του φορτίου στις μονάδες παραγωγής γνωστό ως Economic Power Dispatch (ρύθμιση δηλαδή των επιμέρους ισχύων εξόδου) όταν το κόστος παραγωγής στην εκάστοτε δομή δεν ακολουθεί γραμμική σχέση.

2. Οι μονάδες παραγωγής

2.1 Η δομή

Πέρα των νεοεισαχθέντων φωτοβολταϊκών εγκαταστάσεων και χημικών συσσωρευτών , κάθε εργαστάσιο ή μικρής κλίμακας παραγωγός αξιοποιεί κάποια μορφή παραγωγής μηχανικής ισχύος (ατμοστρόβιλοι, μεικτού κύκλου θεμρικές μηχανές, φτερωτή υδροηλεκτρικού εργοστασίου κ.α.) και μια ηλεκτρογεννήτρια της οποίας η έξοδος μπορεί να ρυθμιστεί επηρρεάζοντας παράγοντες της (όπως το ρεύμα διέγερσης, ηλεκτρονικοί μετατροπείς στην έξοδο κ.α.).

Ανάλογα τον τύπο των εγκαταστάσεων, θερμοδυναμικούς, μηχανικούς και ηλεκτρικούς περιορισμούς η **παρεχόμενη ισχύς** παρουσιάζει **ανώ και κάτω όρια** τα οποία συσχετίζονται με την αντοχή και την καλή λειτουργεία της δομής. Συνήθως όσο μεγαλύτερη η ονομαστική ισχύς τόσο μικρότερη και η ευελιξία της μονάδας. Ένας ακόμα παράγοντας είναι η ενεργοποίηση δευτερευόντων **βοηθητικών συστημάτων** (π.χ. προθερμαντήρας ατμού σε μεικτού κύκλου εγκαταστάσεις) τα οποία τροποποιούν τα χαρακτηριστικά της απόδωσης αλλά μπορούν να ενεργοποιηθούν μόνο πάνω από κάποια δεδομένη στάθμη λειτουργείας.

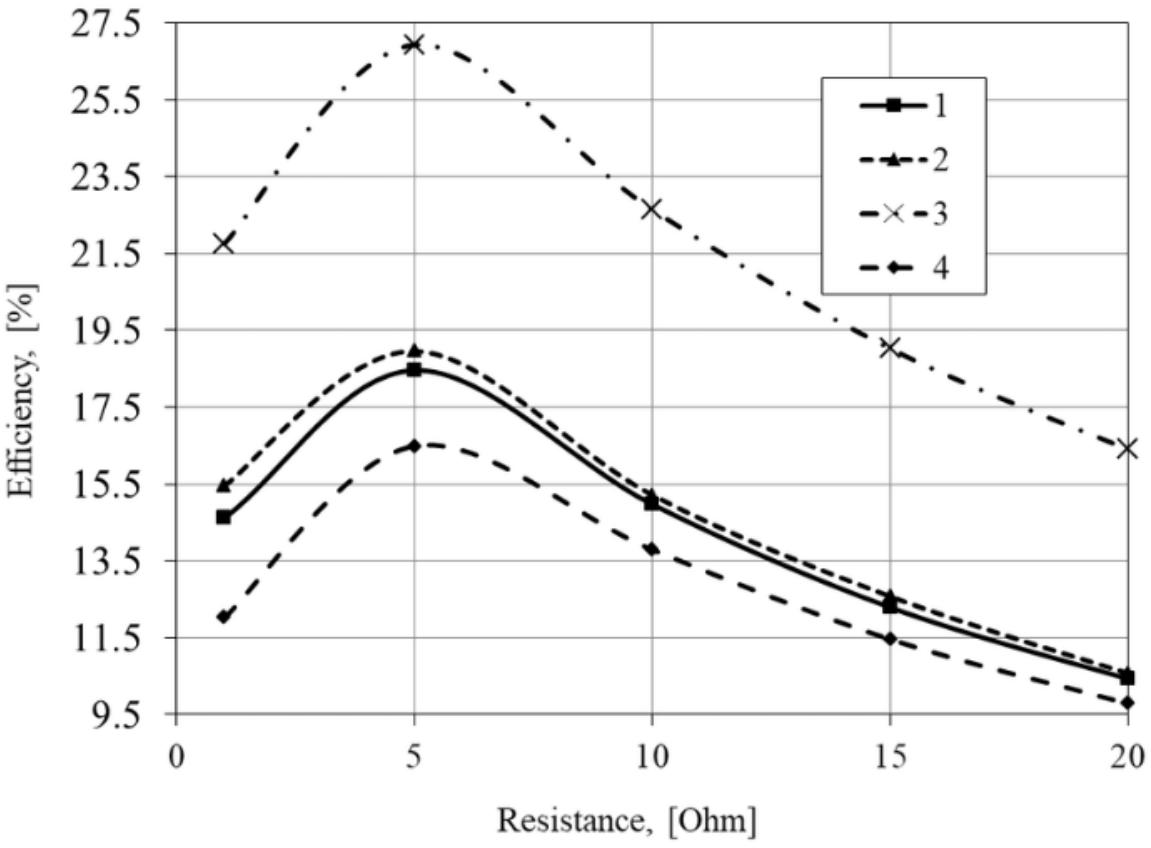
Πολύ συχνά ένας σταθμός μπορεί να **περιλαμβάνει πάνω από ένα υπο-σύστημα** και δύναται να ελέγχει ξεχωριστά την παραγωγή σε κάθε ένα από αυτά. Ο διαχωρισμός αυτός επιτρέπει μεγαλύτερη **ευελιξία** στη ρύθμιση της παραγωμενης ισχύς **απενεργοποιώντας και ενεργοποιώντας επιμέρους υπομονάδες** ενώ ταυτόχρονα προσφέρει μικρότερο κόστος κατασκευής.

Υδροηλεκτρικό εργοστάσιο με πολλαπλές φτερωτές

2.2 Προέλευση σχέσης ισχύος-κόστους

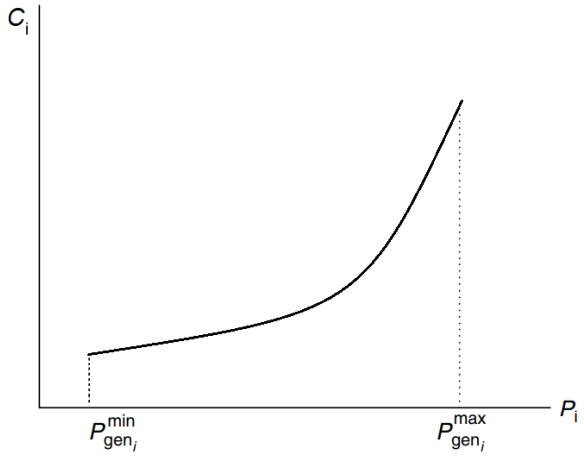
Σε καποιες περιπτώσεις τα όρια λειτουργείας περιορίζουν την εφικτή περιοχή λειτουργείας σε ένα μικρό διάστημα και η θεώρηση πως το κόστος ανα MWh είναι σταθερό έχει υπόσταση. Παρόλα αυτά, φαινόμενα φυσικά και λογιστικά επηρρεάζουν την ευθεία αυτή και έτσι όσο μεγαλύτερο τμήμα της εφικτής λειτουργείας εξετάζουμε τόσο αποκλίνουμε από αυτήν την θεώρηση.

Η **ηλεκτρομηχανική απόδωση** των γεννητριών καθώς και συνοδευτικών ηλεκτρονικών στοιχείων, όπου αυτά υπάρχουν, συνήθως ακολουθεί μία καμπύλη ανάστροφης και ασύμετρης καμπάνας όπως φαίνεται στο παρακάτω σχήμα. (Να σημειωθεί πως η αντίσταση του φορτίου υποδηλώνει την απορροφώμενη ισχύ σε αντίστροφη αναλογία καθώς $P=V^2/R$ με V=σταθερή τάση δικτύου)

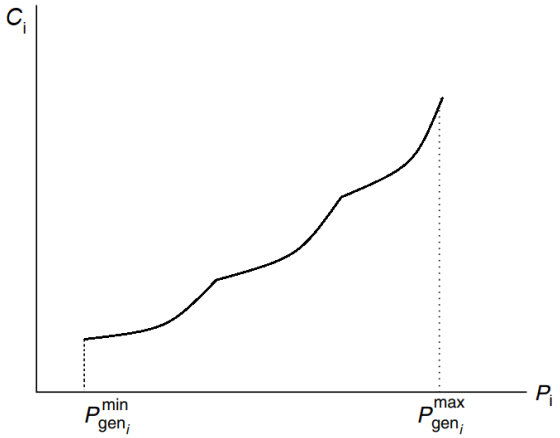


Καμπύλες απόδωσης 4 διαφορετικών γεννητριών

Ακόμα, **λογιστικά μοτίβα** όπως οι οικονομίες κλίμακως, τα κόστη συντήρησης (που εξαρτόνται από την στάθμη λειτουργείας σε βάθος χρόνου) και η ακαμψία της τροφοδοτικής αλυσίδας καυσίμων δημιουργούν επιπλέον μη γραμμικούς όρους οι οποίοι στοχαστικά φέρνουν την καμπύλη κόστους μιας μονάδας στην παρακάτω γενική μορφή.



Γενική μορφή



Παράδειγμα Valve-Point effect στην καμπύλη κόστους

Ειδικότερα αν μία μονάδα αποτελείται από **συνδιασμό μικροτέρων** τότε η ολική καμπύλη για το εργοστάσιο έχει επιπλέον παραμορφώσεις που αντιστοιχούν στην ενεργοποίση νέων υποσυστημάτων όσο αυξάνεται η ζήτηση. Παραμορφώσεις δημιουργούνται και από την ενεργοποίηση βοηθητικού εξοπλισμού που μπορεί να λειτουργήσει μόνο πάνω από κάποια στάθμη ισχύος ή όπου η αύξηση της ισχύος γίνεται σε ημι-ακέραια βήματα (π.χ. **valve-point effect**).

3.Μια σύνοψη των έως τώρα μεθόδων

3.1 Χρήση πλήρως γραμμικού μοντέλου

Η προφανής, γρήγορη και συχνά επαρκής μέθοδος είναι η θεώρηση μιας μέσης τιμής κόστους ανά MWh σε κάθε σταθμό (μέση κλίση καμπύλης κόστους). Το μοντέλο αυτό μπορεί να εκφραστεί ως ένα LP πρόβλημα όπου οι μεταβλητές Xi αντιστοιχούν στην ισχύ σε κάθε μονάδα και η αντικειμενική συνάρτηση είναι το άθροισμα των γινομένων των Xi με τα αντίστοιχα κόστη Ci. Ο μόνος περιορισμός είναι πως το άθροισμα των Xi πρέπει να ισούτε με την ζήτηση από το δίκτυο Q και τα όρια λειτουργείας για κάθε μονάδα εμφανίζοται ως περιορισμοί στις αντίστοιχες Xi.

Για αυτή τη θεώρηση μάλιστα ο LP συχνά δεν είναι η βέλτιστη λύση και αλγορυθμικά είναι ταχύτερο να τοποθετηθουν σε έναν ταξινομημένο πίνακα (με βάση το κόστος) όλες οι μονάδες με τους περιορισμούς λειτουργείας τους και καθώς προσπελνάμε τον πίνακα από φθηνές σε ακρειβές δομές ενεργοποιούμε τις αντίστοιχες μονάδες. Για αυτήν την επίλυση θεωρείται πως το κάτω όριο των μονάδων είναι το 0. Αν και μη πλήρως ρεαλιστική αυτή η μέθοδος είναι συχνά επαρκής και αξιοποιείται λόγω της ταχύτητας υπολογισμού της σε συχνές ανανεώσεις της ζήτησης.

3.2 Μη γραμμικός προγραμματισμός

Αξίζει να γίνει και μία σύντομη αναφορά στη χρήση μη γραμμικού προγραμματισμού για την επίλυση του Economic Dispatch η οποία αξιοποιείται όταν οι αποκλίσεις από τα LP μοντέλα δημιουργούν σημαντικές απώλειες. Αλγόρυθμοι όπως Particle Swarm και ADMM είναι κάποιες από τις μεθόδους που αξιοποιούνται. Καθώς ο μη γραμμικός προγραμματισμός δεν είναι στα αντικείμενα του μαθήματος περισσότερες πληροφορίες υπάρχουν στη βιβλιογραφεία.

3.3 Τμηματική προσέγγιση με παραδοχές

Μια μέθοδος πιο κοντά στο μοντέλο που καταλήγω είναι η piece-wise προσέγγιση του προβλήματος με χρήση LP. Αυτή η μέθοδος **περιορίζεται σε καμπύλες όπου η δεύτερη παράγωγος είναι μεγαλύτερη ή ίση του μηδενός**. Αυτός ο περιορισμός αν και μας επιτρέπει πλήρη επίλυση με simplex δεν καλύπτει διάφορες μορφές καμπυλών. Ακολουθεί η εξήγηση της μεθόδου αυτής από το βιβλίο 'Power Generation Operation and Control' :

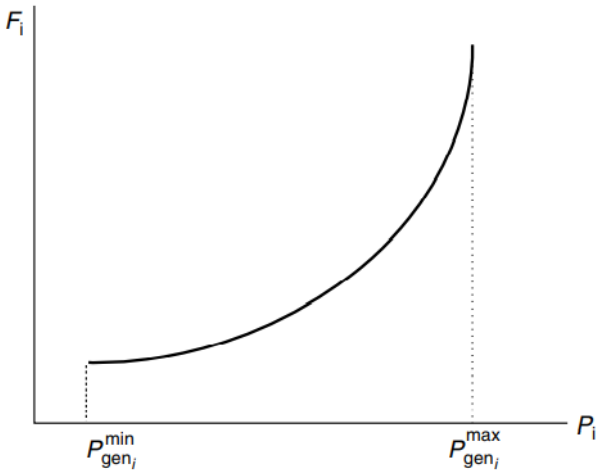


FIGURE 3.3 Nonlinear cost function characteristic.

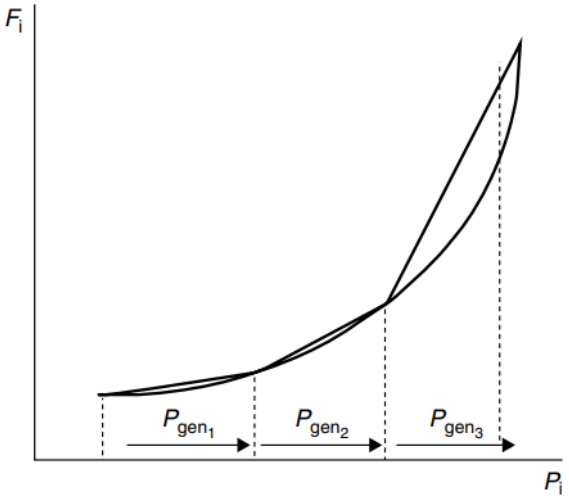


FIGURE 3.4 Nonlinear cost function approximated by straight-line segments.

We start with a nonlinear cost function shown in Figure 3.3.

We can approximate this nonlinear function as a series of straight-line segments as shown in Figure 3.4.

The three segments for generator i shown will be represented as $i1$, $i2$, and $i3$. The P_i variable is replaced with three new variables $P_{\text{gen}_{i1}}, P_{\text{gen}_{i2}}$, and $P_{\text{gen}_{i3}}$. Each segment will have a slope designated s_{i1}, s_{i2}, s_{i3} (where $s_{i1} < s_{i2} < s_{i3}$); then the cost function itself is now represented as the sum of the cost at $P_i^{\min}$ plus the sum of the linear cost for each segment which is simply its slope times the P_{ij} variable. Then

$$F_i\left(P_{\text{gen}_i}\right) = F_i\left(P_{\text{gen}_i}^{\min}\right) + s_{i1}P_{\text{gen}_{i1}} + s_{i2}P_{\text{gen}_{i2}} + s_{i3}P_{\text{gen}_{i3}}$$

$$0 \le P_{\text{gen}_{ik}} \le P_{\text{gen}_{ik}}^{\min} \quad \text{for} \quad k = 1,2,3$$

$$P_{\text{gen}_i} = P_{\text{gen}_i}^{\min} + P_{\text{gen}_i} + P_{\text{gen}_{i1}} + P_{\text{gen}_{i2}} + P_{\text{gen}_{i3}}$$

$$S_{ik} = \frac{F_i\left(P_{\text{gen}_{ik+1}}\right) - F_i\left(P_{\text{gen}_{ik}}\right)}{\left(P_{\text{gen}_{ik+1}}\right) - \left(P_{\text{gen}_{ik}}\right)}$$

The cost function is now made up of a linear expression in the three variables $P_{\text{gen}_{i1}}, P_{\text{gen}_{i2}}, P_{\text{gen}_{i3}}$.

Because the slopes increase in value, the linear program will cause $P_{\text{gen}_{ik}}$ to be at its limit $P_{\text{gen}_{ik}}^{\max}$ before $P_{\text{gen}_{i(k+1)}}$ increases beyond 0.

Σαν συμπέρασμα μπορούμε να πούμε πως αυτή η μέθοδος αξιοποιεί ένα ‘κόλπο’ για να αποφύγει τη χρήση mixed integer linear programming. Πρακτικά εκμεταλλεύεται το γεγονός πως αν η δεύτερη παράγωγος είναι θετική για κάθε κλίση-τμήματος Si θα ισχύει Si<Si+1. Έτσι η αντικειμενική συνάρτηση εξασφαλίζει τη συνθήκη πως θα πρέπει να εξαντληθούν τα όρια της μεταβλητής Pi πρίν γίνει μη μηδενική η Pi+1. Έτσι η μέθοδος ανάγεται στην επίλυση με πλήρως γραμμικό μοντέλο και μπορούν να αξιοποιηθούν και οι προαναφερθείσες βελτιστοποιήσεις έναντι simplex.

4. MILP τμηματική μοντελοποίηση

Στα πλαίσια της εργασίας θα εφαρμόσουμε μία τεχνική παρόμοια με αυτή από την παράγραφο 3.3 αλλά θα επιχειρήσουμε να την γενικεύσουμε και σε καμπύλες που δεν τηρούν την προυπόθεση για την δεύτερη παράγωγο αξιοποιώντας μεικτό γραμμικό προγραμματισμό.

4.1 Δεδομένα

Η πληροφορία για την καμπύλη κόστους δεν μας δίνεται σε αναλυτική μορφή αλλά ώς ένα σετ κλίσεων της καμπύλης για διαστήματα ισχύος και μίας τιμής εκκίνησης. Τα διαστήματα καθώς και το πλήθος των υποδιαιρέσεων διαφέρουν ανά μονάδα οπότε η ακρίβεια και πολυπλοκότητα του μοντέλου θα εξαρτάται από την περιγραφή του κόστους που μας παρέχεται. Κάθε υποδιαίρεση της καμπύλης αυξάνει την διάσταση του μοντέλου κατά μια γραμμική και μια ακέραιη μεταβλητή όπως θα δούμε παρακάτω.

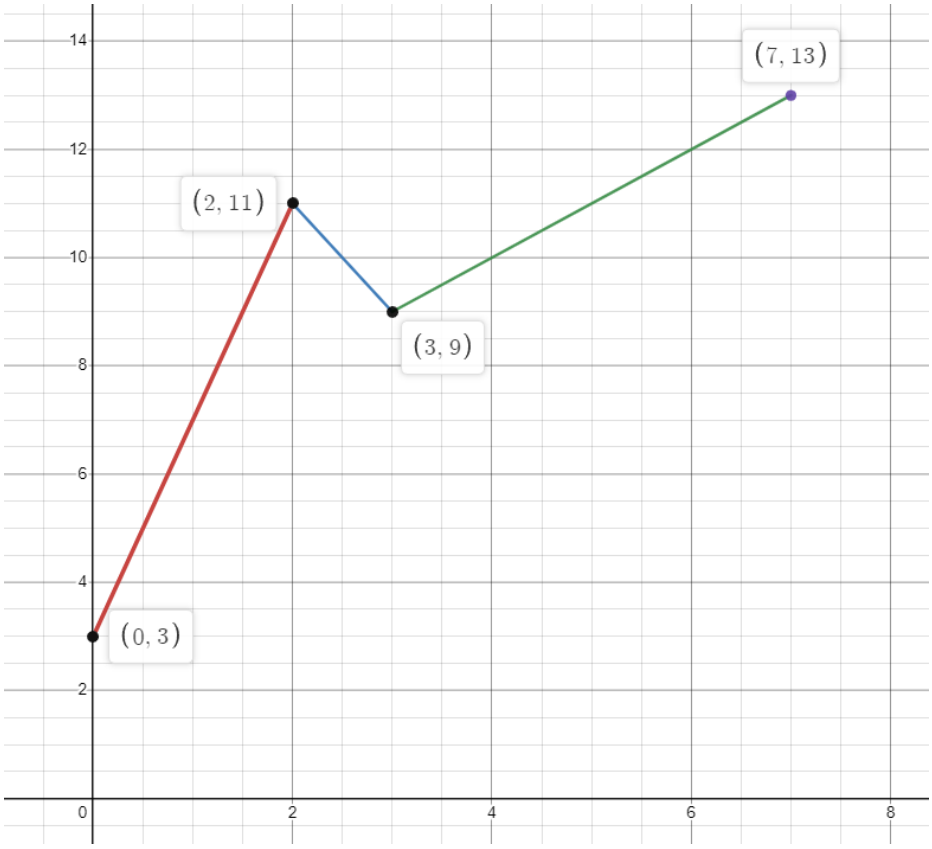
Κατά την ανάπτυξη του μοντέλου θα θεωρήσουμε τα παρακάτω δεδομένα για μια **φανταστική μονάδα παραγωγής** τα οποία **δεν ανταποκρίνονται** απαραίτητα **στη πραγματικότητα** αλλά καλύπτουν τις **ανάγκες επίδειξης της μεθόδου**. Στην παράγραφο 4.3 η μέθοδος θα γενικευτεί και στην ενότητα 5 θα αξιοποιήσουμε ρεαλιστικά ψευτοδεδομένα.

Τιμή εκκίνησης 3k$

Διάστημα Ισχύος (MWh)	Κόστος (k$) ανά MWh
0-2	4
2-3	-2
3-7	1

Έχόντας τα παραπάνω δεδομένα μπορούμε να κατασκευάσουμε μια κλαδωτή συνάρτηση Z για την καμπύλη κόστους με μεταβλητή την X.

$$z = \begin{cases} 3 + 4x & \text{for } 0 \leq x \leq 2 \\ 15 - 2x & \text{for } 2 \leq x \leq 3 \\ 6 + x & \text{for } 3 \leq x \leq 7. \end{cases}$$



4.2 Μετατροπή πολυκλαδικής συνάρτησης

Αν και η συνάρτηση κόστους αποτελείται από πολλές μικρότερες γραμμικές συναρτήσεις η ίδια δεν είναι γραμμική και δεν μπορεί να εισαχθεί σε μοντέλο γραμμικού προγραμματισμού.

Θα ακολουθήσουμε λοιπόν μια διαδικασία γραμμικοποίησης ή οποία στην περίπτωση που η δεύτερη παράγωγος ήταν θετική θα μπορούσε να αναχθεί στη μέθοδο από το βιβλίο ‘Power Generation

Operation and Control' όπως στην παραγράφο 3.3 (με κάποιες αλλαγές) αλλά εδώ θα καλύψουμε όλες τις δυνατές μορφές της καμπύλης κόστους.

Αρχικά εισάγουμε 4 νέες μεταβλητές (n+1 όπου n τα διαστήματα ισχύος) X1,X2,X3,X4. Θέτουμε X=Σ(ki*Xi) όπου ki είναι οι τιμές της X στα άκρα των διαστημάτων ισχύος. Άρα για το παράδειγμά μας :

$$x = 0x_1 + 2x_2 + 3x_3 + 7x_4$$

Ακόμα απαιτούμε Σ(Xi)=1, ακόμα για τα Xi θέλουμε να απαιτήσουμε πως μόνο 2 από αυτά και μόνο αν είναι διαδοχικά μπορούν να είναι μη μηδενικά. Αυτό θα το υλοποιήσουμε όμως στο τέλος με ακέραιο προγραμματισμό οπότε προς το παρόν το αγνοούμε.

Έχοντας τις παραπάνω προυποθέσεις μπορούμε να ορίσουμε την συνάρτηση Z, την καμπύλη κόστους δηλαδή, ως Z=Σ(di*Xi) όπου di είναι οι τιμές της Z στα άκρα των διαστημάτων ισχύος. Άρα για το παράδειγμά μας :

$$z = 3x_1 + 11x_2 + 9x_3 + 13x_4$$

Ακόμα πρέπει να εξασφαλίσουμε πως μόνο δύο και μόνο διαδοχικές μεταβλητές μπορούν να είναι διάφορες του μηδενός, π.χ. για **X_1=0.5 και X_2=0.5** (άρα Z=3*0.5+11*0.5+0+0=1.5+5.5) **Z=7** και (X=0.5*0+0.5*2+0+0) **X=1** άρα **Z(1)=7** που ισχύει ενώ για **X_1=0.5 και X_3=0.5** **X=1.5** και **Z=6** ενώ **Z(1.5)=9**. Για να υλοποιηθεί αυτό θα εισάγουμε έναν νέο περιορισμό 'SOS-2 constraint' (special order set) ο οποίος αξιοποιεί ακέραιες μεταβλητές για να εξασφαλίσει την διαδοχικότητα των μη μηδενικών μεταβλητών.

Θέτουμε λοιπόν 4 νέες δυαδικές μεταβλητές Yi και απαιτούμε Xi≤Yi, οπότε άν μία μεταβλητή Yi είναι 1 τότε μόνο η αντίστοιχη Xi μπορεί να είναι μη μηδενική. Ακόμα θέτουμε Σ(Yi) ≤ 2 ώστε να υπάρχουν έως 2 μεταβλητές Yi (και αντίστοιχα Xi) μη μηδενικές ταυτόχρονα και τέλος θέτουμε πως τα επιμέρους αθροίσματα της Yi με τις υπόλοιπες Y πέρα των Yi-1 και Yi+1 θα είναι μικρότερα ή ίσα του 1 (άρα 0). Οπότε για το παράδειγμά μας :

$$
\begin{array}{rcll}
x_i & \leq & y_i & i = 1, \ldots, 4 \\
\sum_{i=1}^{4} y_i & \leq & 2 & \\
y_1 + y_3 & \leq & 1 & \\
y_1 + y_4 & \leq & 1 & \\
y_2 + y_4 & \leq & 1. &
\end{array}
$$

4.3 Γενική μορφή

Τέλος θα ανάξουμε το παραπάνω μοντέλο σε μία γενική μορφή πριν το εφαρμώσουμε στον κώδικα.

Για <u>κάθε μονάδα παραγωγής 'j'</u> έχουμε τα ακόλουθα:

<u>Δυαδικές</u> : Y_{ji}

<u>Μη αρνητικές</u>: X_{ji}, X_j, Z_j

<u>Παράμετροι</u>: k_{ji}, d_{ji}, n_j

<u>Περιορισμοί:</u>

$\Sigma(X_{ji})=1$

$X_j=\Sigma(k_{ji}*X_{ji})$

$Z_j=\Sigma(d_{ji}*X_{ji})$

$X_{ji} \leq Y_{ji}$

$\Sigma(Y_{ji}) \leq 2$

$Y_{ji}+Y_{jr} \leq 1$ όπου r ανήκει στο [1,n] εκτός των i+1 και i-1 (άν υπάρχουν)

Οι περιορισμοί άνω και κάτω ορίου λειτουργείας τελικά απορροφήθηκαν κατά τη μετατροπή της συνάρτησης από κλαδωτή εντός των περιορισμών $\Sigma(X_{ji})=1$, $X_j=\Sigma(k_{ji}*X_{ji})$ και της θετικότητας των X_{ij}

Και τέλος:

<u>Αντικειμενική Συνάρτηση</u>: $\Sigma(Z_j)$

<u>Βελτιστοποίηση</u>: Min

<u>Περιορισμός</u>: $\Sigma(X_j)=Q$

5. Σύσταση εφαρμογής

5.1 Μεταφορά μοντέλου σε περιβάλλον python

Αξιοποιώντας την βιβλιοθήκη pulp και το περιβάλλον της python κατασκευάζουμε την εφαρμογή. Τα δεδομένα φορτώνονται από αρχείο .txt της μορφής :

data.txt

```
0,2,3,7
3,11,9,13
0
&
4,7,10
1,14,95
1
&
0,1,2,3,4,5,6,7,8,9,10,11,12,13
2,2.5,3,4,7,15,16,17,19,20,22,25,34,39
3
```

Τα δεδομένα της κάθε γεννήτριας είναι χωρισμένα με το χαρακτήρα '&' και αποτελούνται απο 3 γραμμές , η πρώτη είναι οι διάφορες τιμές της ισχύος όπου γίνεται αλλαγή τμήματος, η δεύτερη είναι η τιμής της καμπύλης στις παραπάνω θέσεις και η τρίτη περιέχει ένα σειριακό αριθμό για την αρίθμιση των μονάδων.

Για κάθε μονάδα κατασκευάζουμε μια X (ισχύς) μεταβλητή και μια Z (κόστος), και οι δύο είναι συνεχείς και μη αρνητικές :

```
for data_set in data:
    j=data_set[2]
    temp=LpVariable('Z_{}'.format(j), lowBound=0, upBound=None, cat='Continuous')#Z variable
    variables['Z_j={}'.format(j)]=temp
    Z.append(temp)
    temp=LpVariable('X_{}'.format(j), lowBound=0, upBound=None, cat='Continuous')#X variable
    variables['X_j={}'.format(j)]=temp
    X.append(temp)
```

Έχοντας τις μεταβλητές Z και X ορίζουμε το πρόβλημα, την αντικειμενική συνάρτηση και τον πρώτο περιορισμό (Σ(Xj)=Q).

```
problem = LpProblem("Project_gram_sund", LpMinimize)
#Σ(Zj)
problem += lpSum(Z)
#Σ(Xj)=Q
problem += lpSum(X)==Q
```

Και εντός της συνάρτησης 'add_generator' που τρέχει μία φορά για κάθε γεννήτρια κατασκευάζουμε μία x (συνεχή μεταξύ 0 και 1) και μία y (δυαδική) μεταβλητή για κάθε data point :

```
for i in range(0,n):
    temp=(LpVariable("y_{}_{}".format(j,i+1), lowBound=0, upBound=1, cat='Binary'))#y_variables (binary)
    variables['y_j={}_i={}'.format(j,i+1)]=temp
    y.append(temp)
    temp=(LpVariable("x_{}_{}".format(j,i+1), lowBound=0, upBound=1, cat='Continuous'))#x variables (non negative)
    variables['x_j={}_i={}'.format(j,i+1)]=temp
    x.append(temp)
```

Τέλος εισάγουμε τους περιορισμούς για κάθε γεννήτρια στο πρόβλημα:

```
#Xji≤Yji
    for i in range(0,n):
        problem += x[i]-y[i]<=0

#Σ(Xji)=1
    problem += lpSum(x)==1

#Σ(Yji) ≤ 2
    problem += lpSum(y)<=2

#Xj=Σ(kji*Xji)
    problem += lpSum([x_and_X[i]*k[i] for i in range(0,n+1)])==0

#Zj=Σ(dji*Xji)
    problem += lpSum([x_and_Z[i]*d[i] for i in range(0,n+1)])==0

#Yji+Yjr ≤ 1
    for i in range(0,n):
        for m in range(0,n):
            if abs(m-i)>1:
                problem += y[i]+y[m]<=1 #imperfection we get double the rest
```

5.2 Πρόβλημα και λύση scalability

Κατά τον ορισμό του περιορισμού $Yji+Yjr \leq 1$ παρατηρούμε πως οι γραμμές που πρέπει να προσθέσουμε στο πρόβλημα σε σχέση με των αριθμό των data points είναι **$O(n^2)$,** πιο συγκεκριμένα n^2-2n καθώς πρέπει για την κάθε Yi να προσθέσουμε μια γραμμή για κάθε άλλη μεταβλητή πέρα των Y_{i-1} και Y_{i+1} . Αυτό το πρόβλημα διογκώνεται και επιβραδύνει την λύση όσο μεγαλώνει η απαιτούμενη ακρίβεια της μοντελοποίησης των καμπυλών κόστους. Με παράδειγμα τα δεδομένα από τη φωτογραφία στην παράγραφο 5.1 (1 γεννήτρια με 3 data points, μία με 4 και μία με 14) δημιουργούνται **~210 γραμμές** περιορισμών.

Θέλοντας να είναι δυνατή η μοντελοποίηση με π.χ. εκατό σημεία ανά καμπύλη δημιουργούνται περί των 5000 περιορισμών ανά γεννήτρια. Η λύση σε αυτό το πρόβλημα είναι να παρεκλίνουμε από τον κλασικό ορισμό των SOS-2 περιορισμών και για κάθε Yi μπορούμε να ορίσουμε 2 περιορισμούς. Ο ένας είναι το άθροισμα της Yi με όλες της ζυγές Ys όπου s>i+1 να είναι μικρότερο ή ίσο του 1 και ο δεύτερος είναι το ίδιο αλλά με τις περριτές. Το αποτέλεσμα τελικά είναι το ίδιο καθώς εξασφαλίζεται πως μόνο διαδοχικές μεταβλητές Y μπορούν να είναι μη μηδενικές αλλά προσθέτουμε μόνο 2n γραμμές οπότε καταλήγουμε σε **O(n)** πολυπλοκότητα και για το παραπάνω παράδειγμα από την παράγραφο 5.1 έχουμε **~70 γραμμές** συνολικά. Στο παράδειγμα των 100 datapoints θα είχαμε περίπου 250 γραμμές ανά γεννήτρια έναντι 5000.

Τα δύο αρχεία .py που παραδίδω διαφέρουν ως προς αυτή τη βελτιστοποίηση που πρόσθεσα , ακολουθεί ο σχετικός

εναλλακτικός κώδικας για τον ορισμό του προαναφερθέντος περιορισμού:

```
#Yji+Yjr ≤ 1
    print(y_evens, y_odds)
    del y_evens[0]
    for i in range(0,n-1):
        if (i % 2)==1:
            del y_evens[0]

        if (i % 2)==0:
            del y_odds[0]
        print(y_evens,y_odds)
        if len(y_odds)>0:
            problem += lpSum(y_odds) + y[i] <=1
        if len(y_evens)>0:
            problem += lpSum(y_evens) + y[i] <=1
```

5.3 Προσομοίωση και αποτελέσματα

Τέλος μπορούμε να αξιοποιήσουμε την .solve() μέθοδο και να πάρουμε το τελικά ελάχιστο κόστος καθώς και τις επιμέρους στάθμες ισχύος για κάθε εργοστάσιο:

```
Result - Optimal solution found

Objective value:                6.50000000
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             0.01
Time (Wallclock seconds):       0.01

Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.02   (Wallclock seconds):       0.02

X_0 = 0.0
X_1 = 4.0
X_2 = 1.0
Z_0 = 3.0
Z_1 = 1.0
Z_2 = 2.5
```

Αποτελέσματα για το παράδειγμα της εικόνας με τα δεδομένα από την παράγραφο 5.1

Παράδειγμα εσωτερικής δομής προβλήματος:

```
\* Project_gram_sund *\
Minimize
OBJ: Z_0
Subject To
_C1: X_0 = 5
_C10: y_0_1 + y_0_4 <= 1
_C11: y_0_1 + y_0_3 <= 1
_C12: y_0_2 + y_0_4 <= 1
_C2: x_0_1 - y_0_1 <= 0
_C3: x_0_2 - y_0_2 <= 0
_C4: x_0_3 - y_0_3 <= 0
_C5: x_0_4 - y_0_4 <= 0
_C6: x_0_1 + x_0_2 + x_0_3 + x_0_4 = 1
_C7: y_0_1 + y_0_2 + y_0_3 + y_0_4 <= 2
_C8: - X_0 + 2 x_0_2 + 3 x_0_3 + 7 x_0_4 = 0
_C9: - Z_0 + 3 x_0_1 + 11 x_0_2 + 9 x_0_3 + 13 x_0_4 = 0
Bounds
 x_0_1 <= 1
 x_0_2 <= 1
 x_0_3 <= 1
 x_0_4 <= 1
Binaries
y_0_1
y_0_2
y_0_3
y_0_4
End
```

Δομή προβλήματος για την περίπτωση μίας γεννήτριας με δεδομένα από το παράδειγμα της ενότητας 4.

6. Σύνοψη

Εν τέλη η προσέγγιση που υλοποιήθηκε έχει εμφανή πλεονεκτήματα έναντι της απλής γραμμικής θεώρησης και μπορεί να μοντελοποιήσει περισσότερα προβλήματα από την

μέθοδο της παραγράφου 3.3. Ταυτόχρονα δεν απαιτεί τους πόρους ενός μη γραμμικού επιλυτή και εγγυάται πως η λύση θα είναι βέλτιστη. Ο υπολογιστικός φόρτος είναι πιο κοντά στις υπάρχουσες LP μεθόδους μιας και χρησημοποιούμε της ίδιας τάξεις στήλες πίνακα (εξαρτώνται αναλογικά από τον αριθμό γεννητριών και την ακρίβεια) και ενώ οι γραμμές είναι περισσότερες το πρόβλημα ακολουθεί O(n) τάξη αύξησης γραμμών με βάση την ακρίβεια.

7. Βιβλιογραφεία

LP & EDL

Valve Point Effect

SOS-2 piecewise Μέθοδος από τον John Mitchel

‘Power Generation Operation and Control’ 3d edition

Generator efficiency

Non linear solver

Non linear solver considering valve point effect