Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Τεχνικές Βελτιστοποίησης

Εργασία

Ελαχιστοποίηση συνάρτησης πολλών μεταβλητών – Γενετικοί αλγόριθμοι

Σκλαβενίτης Γεώργιος
ΑΕΜ: 10708

Χειμερινό Εξάμηνο 2024-2025

Περιεχόμενα

1 Μαθηματική περιγραφή του προβλήματος

Εισαγωγή στους Γενετικούς Αλγορίθμους
Οι γενετικοί αλγόριθμοι είναι μέθοδοι βελτιστοποίησης εμπνευσμένες από τη φυσική εξέλιξη. Οι βασικές αρχές τους περιλαμβάνουν:

- Επιλογή (Selection): Κρατούν τις καλύτερες λύσεις από έναν πληθυσμό.
- Διασταύρωση (Crossover): Συνδυάζουν στοιχεία από δύο *γονείς* για να δημιουργήσουν νέες λύσεις *(απογόνους)*.
- Μετάλλαξη (Mutation): Τροποποιούν τυχαία μέρη μιας λύσης για να εξερευνήσουν νέες δυνατότητες.

Ο στόχος είναι να βρεθεί μια λύση που **ελαχιστοποιεί** ή **μεγιστοποιεί** μια συνάρτηση (fitness function) μέσω επαναλήψεων.

Το Πρόβλημα που Επιλύουμε
Το δίκτυο αποτελείται από 17 ακμές (δρόμους), καθεμία με τα εξής χαρακτηριστικά:

- Μέγιστη δυνατότητα διέλευσης c_i.
- Ρυθμός διέλευσης x_i, ο οποίος πρέπει να υπολογιστεί.

Ο στόχος είναι να ελαχιστοποιηθεί ο συνολικός χρόνος διάσχισης:

$$F(x) = \sum_{i=1}^{17} \left(t_i + \frac{a_i x_i}{1 - \frac{x_i}{c_i}} \right), \tag{1}$$

υπό τους περιορισμούς:

- $\sum x_i = V$: Ο συνολικός ρυθμός διέλευσης πρέπει να είναι ίσος με V.
- $x_i \leq c_i$: Ο ρυθμός σε κάθε ακμή δεν μπορεί να υπερβαίνει τη μέγιστη δυνατότητά της.

Η Εφαρμογή του Γενετικού Αλγορίθμου
Ο γενετικός αλγόριθμος βελτιώνει σταδιακά τις λύσεις $x = [x_1, x_2, \ldots, x_{17}]$ μέσω επαναλήψεων *(γενεών)* για να βρει την καλύτερη δυνατή λύση που ελαχιστοποιεί το $F(x)$. Η λειτουργία του ακολουθεί την παρακάτω δομή:

1. Αρχικοποίηση Πληθυσμού
   - Ξεκινάμε με έναν τυχαίο πληθυσμό λύσεων x (π.χ., 50 λύσεις).
   - Κανονικοποιούμε τις λύσεις ώστε να τηρούν τον περιορισμό $\sum x_i = V$.
2. Υπολογισμός Συνάρτησης Ικανότητας (Fitness Function): Για κάθε λύση x, υπολογίζουμε τον συνολικό χρόνο διάσχισης $F(x)$, ο οποίος είναι ο χρόνος που χρειάζεται για να διασχιστεί το δίκτυο.
3. Επιλογή Γονέων (Selection): Επιλέγουμε τις καλύτερες λύσεις από τον πληθυσμό, δηλαδή αυτές με το μικρότερο $F(x)$, χρησιμοποιώντας τη μέθοδο *roulette wheel selection*.
4. Διασταύρωση (Crossover): Συνδυάζουμε στοιχεία από δύο γονείς για να δημιουργήσουμε νέες λύσεις (απογόνους).

5. Μετάλλαξη (Mutation):Τροποποιούμε τυχαία κάποιες τιμές x_i στις λύσεις για να προσθέσουμε ποικιλία. Μετά από κάθε μετάλλαξη, διασφαλίζεται ότι:

   - $\sum x_i = V$.
   - $x_i \leq c_i$.

6. Ενημέρωση Πληθυσμού: Ο νέος πληθυσμός αποτελείται από τους απογόνους που δημιουργήθηκαν μέσω διασταύρωσης και μετάλλαξης.

7. Έλεγχος Τερματισμού: Ο αλγόριθμος επαναλαμβάνεται για έναν προκαθορισμένο αριθμό γενεών ή μέχρι η λύση να σταθεροποιηθεί.

2 Ο Αλγόριθμος

2.1 Initalize Population

Σε αυτό το βήμα, δημιουργείται ένας αρχικός πληθυσμός λύσεων (άτομα), από τον οποίο θα ξεκινήσει η διαδικασία της εξέλιξης. Οι λύσεις αυτές είναι τυχαίες, ώστε να παρέχεται ποικιλομορφία στον πληθυσμό.
Η συνάρτηση αρχικοποίησης πληθυσμού έχει ως εξής:

- Αρχικοποίηση:
  - Η μεταβλητή `pop_size` καθορίζει το μέγεθος του πληθυσμού.
  - Η μεταβλητή `num_edges` είναι ο αριθμός των ακμών στο δίκτυο.
  - Η μεταβλητή `V` είναι ο ρυθμός εισερχομένων οχημάτων.
  - Ο πίνακας `c` περιέχει τις μέγιστες δυνατότητες για κάθε ακμή.
- Δημιουργία Πίνακα Πληθυσμού:
  - Δημιουργείται ένας πίνακας `population` με διαστάσεις `pop_size x num_edges`, αρχικοποιημένος με μηδενικά.
- Γέμισμα του Πληθυσμού με Τυχαίες Λύσεις:
  - Για κάθε άτομο (`i`):
    * Δημιουργείται ένας πίνακας `x` με τυχαίες τιμές για κάθε ακμή, οι οποίες πολλαπλασιάζονται με τις μέγιστες δυνατότητες `c`.
    * Αν το άθροισμα των τιμών του πίνακα `x` είναι μεγαλύτερο από `V`, τότε οι τιμές του `x` κανονικοποιούνται ώστε να είναι ίσες με `V`.
    * Οι τιμές του πίνακα `x` τελικά κανονικοποιούνται ώστε το άθροισμά τους να είναι ίσο με `V`.

2.2 Fitness Function

Η συνάρτηση υπολογίζει μια τιμή καταλληλότητας για κάθε λύση (άτομο) στον πληθυσμό. Οι λύσεις με υψηλότερες τιμές καταλληλότητας είναι πιο πιθανό να επιλεγούν για αναπαραγωγή.
Η διαδικασία καταλληλότητας εκτελεί τα εξής βήματα:

1. Αρχικοποίηση:
   - Ο πίνακας `population` περιέχει όλες τις λύσεις (άτομα) του πληθυσμού.
   - Ο πίνακας `t` περιέχει σταθερούς χρόνους για κάθε ακμή.
   - Ο πίνακας `a` περιέχει συντελεστές για κάθε ακμή.
   - Ο πίνακας `c` περιέχει μέγιστες δυνατότητες για κάθε ακμή.
   - Ο πίνακας `B` περιέχει τις ροές κόμβων.
   - Η μεταβλητή `penalty_factor` είναι ο συντελεστής ποινής.
2. Διάσταση και Αρχικοποίηση Fitness:
   - Με την εντολή `[pop_size, num_edges] = size(population)` λαμβάνουμε το μέγεθος του πίνακα `population`.
   - Δημιουργείται ο πίνακας `fitness` που θα αποθηκεύσει τις τιμές καταλληλότητας για κάθε άτομο του πληθυσμού.
3. Υπολογισμός Fitness για Κάθε Άτομο:

- Για κάθε άτομο (`i`):
  - Ο πίνακας `x` περιέχει τις τιμές του ατόμου.
  - Υπολογίζεται ο χρόνος $T_i(x_i) = t_i + (a_i * x_i)/[1 - (x_i/c_i)]$ για κάθε ακμή.
  - Υπολογίζεται η τιμή κόστους `cost_value = sum( x .* T )`.
  - Υπολογίζεται η ανισορροπία ροής `flow_imbalance = B * x'`.
  - Υπολογίζεται η ποινή `penalty_term = penalty_factor * sum(abs(flow_imbalance))`.
  - Η τελική τιμή καταλληλότητας `fitness(i)` είναι το άθροισμα του κόστους και της ποινής.

2.3 Selection

Η διαδικασία επιλέγει γονείς από τον υπάρχοντα πληθυσμό για να δημιουργήσουν απογόνους μέσω διασταύρωσης και μετάλλαξης. Η επιλογή βασίζεται στην καταλληλότητα των ατόμων (fitness), με στόχο να επιλεγούν τα πιο κατάλληλα άτομα για αναπαραγωγή. Μια δημοφιλής μέθοδος επιλογής είναι η μέθοδος της "ρουλέτας" (roulette wheel selection). Σε αυτήν τη μέθοδο, κάθε άτομο έχει πιθανότητα να επιλεγεί, η οποία είναι ανάλογη με την καταλληλότητά του.
Η διαδικασία επιλογής εκτελεί τα εξής βήματα:

1. Αρχικοποίηση:
   - Ο πίνακας `population` περιέχει τον τρέχοντα πληθυσμό.
   - Ο πίνακας `fitness` περιέχει τις τιμές καταλληλότητας για κάθε άτομο του πληθυσμού.
2. Υπολογισμός πιθανοτήτων επιλογής:
   - Οι πιθανότητες επιλογής υπολογίζονται ως το αντίστροφο του fitness (`prob = 1 ./ fitness`).
   - Οι πιθανότητες κανονικοποιούνται ώστε το άθροισμά τους να είναι 1 (`prob = prob / sum(prob)`).
   - Υπολογίζονται οι σωρευτικές πιθανότητες (`cum_prob = cumsum(prob)`).
3. Διάσταση και δημιουργία γονέων:
   - Με την εντολή `[pop_size, ] = size(population)` λαμβάνεται το μέγεθος του πίνακα `population`.
   - Δημιουργείται ο πίνακας `parents` που θα αποθηκεύσει τους επιλεγμένους γονείς.
4. Διαδικασία Επιλογής:
   - Για κάθε άτομο του πληθυσμού (`i`):
     - Επιλέγεται μια τυχαία τιμή `r` στο διάστημα [0, 1].
     - Βρίσκεται το πρώτο άτομο του πληθυσμού του οποίου η σωρευτική πιθανότητα είναι μεγαλύτερη ή ίση με `r` (`selected_idx = find(cum_prob >= r, 1)`).
     - Το άτομο αυτό προστίθεται στους γονείς (`parents(i, :) = population(selected_idx, :)`).

2.4 Crossover

Η υλοποίηση του αλγορίθμου γίνεται με One-Point Crossover. Αυτό σημαίνει ότι για κάθε ζεύγος γονέων επιλέγεται ένα τυχαίο σημείο διασταύρωσης και οι απόγονοι δημιουργούνται με το να συνδυαστούν τα τμήματα των γονέων πριν και μετά από αυτό το σημείο.
Το crossover γίνεται ως εξής:

- Αρχικοποίηση:
  1. Ο πίνακας parents περιέχει όλα τα γονιδιακά σύνολα του πληθυσμού.
  2. Το crossover_rate είναι η πιθανότητα να γίνει διασταύρωση για κάθε ζεύγος γονέων.
- Διάσταση και δημιουργία απογόνων: Με την εντολή [pop_size, num_edges] = size(parents) παίρνουμε το μέγεθος του πίνακα parents, δηλαδή τον αριθμό των γονέων (pop_size) και τον αριθμό των ακμών ή χαρακτηριστικών (num_edges) και παράλληλα δημιουργούμε τον πίνακα offspring που θα αποθηκεύσει τους απογόνους.
- Διαδικασία Διασταύρωσης: Για κάθε δύο γονείς (i και i+1):
  - Εάν η τυχαία τιμή rand() είναι μικρότερη από το crossover_rate, γίνεται διασταύρωση.
    1. Επιλέγονται δύο γονείς p1 και p2.
    2. Επιλέγεται ένα τυχαίο σημείο διασταύρωσης point με την εντολή randi([1, num_edges-1]).
    3. Δημιουργούνται οι απόγονοι: Ο πρώτος απόγονος (offspring(i, :)) αποτελείται από το τμήμα του p1 μέχρι το σημείο διασταύρωσης και το υπόλοιπο του p2. Ο δεύτερος απόγονος (offspring(i+1, :)) αποτελείται από το τμήμα του p2 μέχρι το σημείο διασταύρωσης και το υπόλοιπο του p1.
  - Εάν η τυχαία τιμή rand() είναι μεγαλύτερη ή ίση με το crossover_rate, οι απόγονοι είναι ακριβή αντίγραφα των γονέων.

2.5 Mutation

Η μετάλλαξη εισάγει τυχαιότητα και ποικιλομορφία στον πληθυσμό. Αυτό γίνεται τροποποιώντας τυχαία ένα ή περισσότερα γονίδια σε έναν απόγονο. Η μετάλλαξη βοηθά στην εξερεύνηση του χώρου αναζήτησης και στην αποφυγή τοπικών ελαχίστων/μεγίστων.
Η διαδικασία μετάλλαξης εκτελεί τα εξής βήματα:

- Αρχικοποίηση:
  - Ο πίνακας `offspring` περιέχει τους απογόνους.
  - Το `mutation_rate` είναι η πιθανότητα μετάλλαξης για κάθε γονίδιο.
  - Ο πίνακας `c` περιέχει τα ανώτατα όρια για τις τιμές των γονιδίων.
  - Η μεταβλητή `V` είναι η συνολική τιμή που πρέπει να διατηρηθεί.
- Διάσταση και αντιγραφή απογόνων:
  - Με την εντολή `[pop_size, num_edges] = size(offspring)` λαμβάνουμε το μέγεθος του πίνακα `offspring`.
  - Δημιουργείται ο πίνακας `mutated`, αντίγραφο του `offspring`.
- Διαδικασία Μετάλλαξης:
  - Για κάθε απόγονο (`i`):
    - Για κάθε γονίδιο (`j`):
      - Εάν η τυχαία τιμή `rand()` είναι μικρότερη από το `mutation_rate`, το γονίδιο μεταλλάσσεται.
      - Το νέο γονίδιο παίρνει μια τυχαία τιμή εντός του εύρους [0, `c(j)`].
- Κανονικοποίηση:
  - Μετά τη μετάλλαξη, κάθε απόγονος κανονικοποιείται ώστε το άθροισμα των γονιδίων του να είναι ίσο με `V`.

2.6 Genetic Algorithm

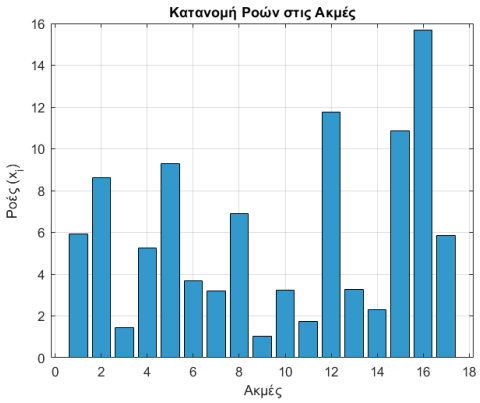
Η διαδικασία του γενετικού αλγορίθμου εκτελεί τα εξής βήματα:

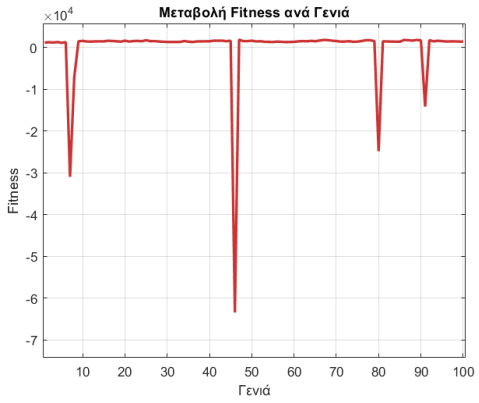
1. Αρχικοποίηση Πληθυσμού:
    - Δημιουργούμε έναν αρχικό πληθυσμό λύσεων χρησιμοποιώντας την συνάρτηση `initialize_population`.
2. Επαναληπτική Διαδικασία Γενετικού Αλγορίθμου:
    - Για κάθε γενιά, εκτελούμε τα εξής βήματα:
        - Υπολογίζουμε την καταλληλότητα (fitness) κάθε ατόμου στον πληθυσμό χρησιμοποιώντας τη συνάρτηση `fitness_function`.
        - Επιλέγουμε γονείς για την επόμενη γενιά χρησιμοποιώντας τη συνάρτηση `selection`.
        - Δημιουργούμε απογόνους μέσω της διασταύρωσης χρησιμοποιώντας τη συνάρτηση `crossover`.
        - Εφαρμόζουμε μετάλλαξη στους απογόνους χρησιμοποιώντας τη συνάρτηση `mutation`.
        - Ανανεώνουμε τον πληθυσμό με τους νέους απογόνους.
3. Εύρεση Τελικής Λύσης:
    - Υπολογίζουμε την καταλληλότητα του τελικού πληθυσμού.
    - Βρίσκουμε το άτομο με την καλύτερη τιμή καταλληλότητας.

3 Αποτελέσματα

3.1 Αρχικός Αλγόριθμος

```
>> genetic_algorithm
  Columns 1 through 8

    5.9090    8.6025    1.4319    5.2503    9.2883    3.6789    3.2105    6.9078

  Columns 9 through 16

    1.0271    3.2310    1.7223   11.7668    3.2700    2.3094   10.8584   15.6990

  Column 17

    5.8367

(Sum(x_i * T_i(x_i)) + penalty): 1406.626994
```



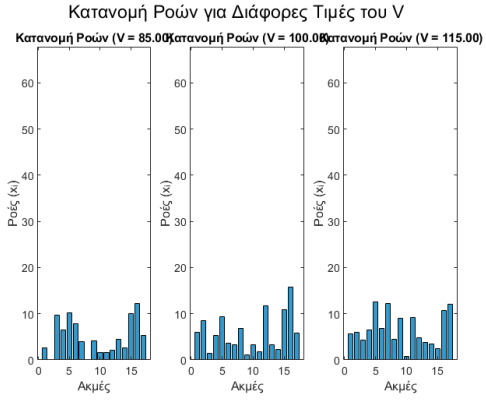


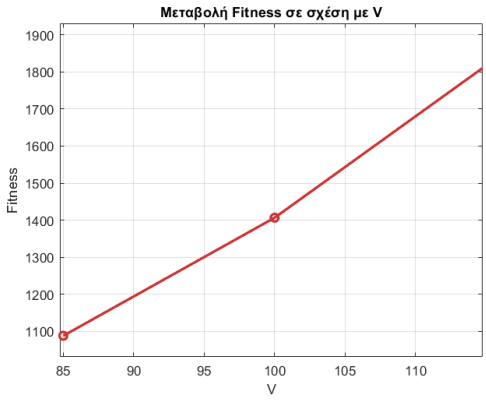
- Η καλύτερη λύση x προσφέρει μια βέλτιστη κατανομή ροών που ελαχιστοποιεί τον συνολικό χρόνο διάσχισης του δικτύου.
- Τα x_i είναι εντός των επιτρεπτών ορίων, ικανοποιώντας τους περιορισμούς $x_i \leq c_i$.
- Ο αλγόριθμος διασφαλίζει την ισορροπία ροών στους κόμβους και τη συνολική ροή $\sum x_i = V$, ενώ δεν υπάρχουν παραβιάσεις δυνατοτήτων.
- Η λύση είναι σταθερή, με ισορροπία ροών και τήρηση περιορισμών, επιβεβαιώνοντας την αξιοπιστία του αλγορίθμου.

3.2 Αλγόριθμος με μεταβολή των εισερχόμενων οχημάτων

```
  >> genetic_algorithm_with_variable_V
For V = 85.00:
  Columns 1 through 8

    2.6315    0.0153    9.6885    6.4273   10.2195    7.8605    3.9860    0.0749

  Columns 9 through 16

    4.1009    1.5873    1.6205    2.0279    4.4993    2.6986   10.0477   12.1531

  Column 17
```

```
    5.3611

  (Sum(x_i * T_i(x_i)) + penalty): 1087.86

For V = 100.00:
  Columns 1 through 8

    5.9090    8.6025    1.4319    5.2503    9.2883    3.6789    3.2105    6.9078

  Columns 9 through 16

    1.0271    3.2310    1.7223   11.7668    3.2700    2.3094   10.8584   15.6990

  Column 17

    5.8367

  (Sum(x_i * T_i(x_i)) + penalty): 1406.63

For V = 115.00:
  Columns 1 through 8

    5.5950    5.9761    4.3266    6.5663   12.6238    6.9144   12.2499    4.5088

  Columns 9 through 16

    8.9770    0.7053    9.2129    4.8671    3.8284    3.4160    2.5071   10.6594

  Column 17

   12.0660

  (Sum(x_i * T_i(x_i)) + penalty): 1817.21
```





Παρατηρούμε ότι το fitness $F(x)$ αυξάνεται με την αύξηση του συνολικού ρυθμού V. Αυτό είναι αναμενόμενο, καθώς μεγαλύτερος ρυθμός οδηγεί σε αυξημένους χρόνους διάσχισης λόγω συμφόρησης. Επίσης,

- Σε χαμηλότερα V (85), οι ροές x_i είναι πιο συγκεντρωμένες σε συγκεκριμένες ακμές.
- Σε υψηλότερα V (115), οι ροές κατανέμονται σε περισσότερες ακμές, με μεγαλύτερες τιμές σε κάποιες από αυτές για να διαχειριστεί η αυξημένη εισροή.

Ο αλγόριθμος προσαρμόζεται επιτυχώς στις διαφορετικές τιμές V, αλλά η συνολική συμφόρηση και ο χρόνος διάσχισης αυξάνονται καθώς αυξάνεται το V.