Τεχνικές Βελτιστοποίησης - Project

Αριστείδης Δασκαλόπουλος (ΑΕΜ: 10640)

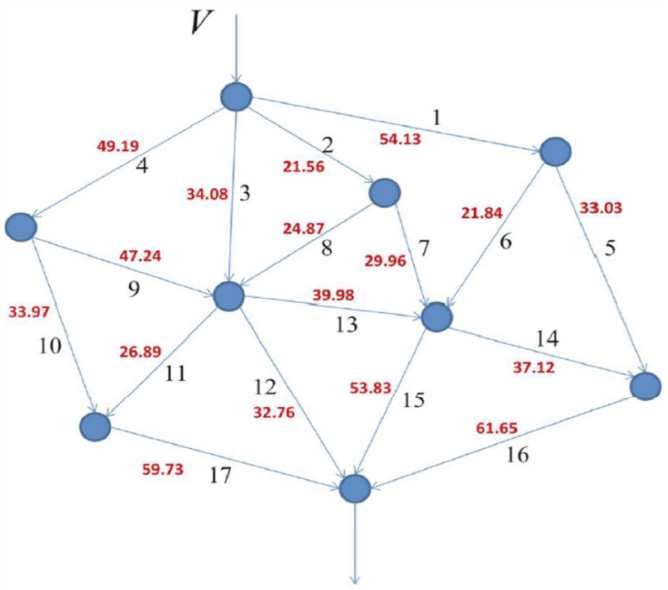
7 Φεβρουαρίου 2025

Εισαγωγή

***Στόχος Εργασίας**: Εύρεση ελαχίστου συνάρτησης πολλών μεταβλητών με χρήση γενετικού αλγορίθμου.*

Το πρόβλημα που καλούμαστε να λύσουμε είναι η ελαχιστοποίηση, ως προς x_i (ρυθμός διέλευσης οχημάτων από τον δρόμο i), του συνολικού χρόνου διάσχισης του παρακάτω δικτύου ανά όχημα - για ρυθμό εισερχόμενων οχημάτων ίσο με V [οχ./min] - με τον περιορισμό ότι όσα οχήματα εισέρχονται σε κάθε κόμβο τόσα και να εξέρχονται:

*ροή οχημάτων που **εισέρχονται** σε έναν κόμβο = ροή οχημάτων που **εξέρχονται** από αυτόν τον κόμβο*



Σχήμα 1: Το οδικό δίκτυο

Για τον δοθέν άκυκλο κατευθυνόμενο γράφο (DAG) έχουμε ότι ο χρόνος κίνησης στον δρόμο/ακμή i, ως συνάρτηση του ρυθμού διέλευσης x_i [οχ./min], δίνεται από:

$$T_i(x_i) = t_i + \frac{a_i x_i}{1 - \frac{x_i}{c_i}} \quad [\text{min}], \quad \text{με: } \lim_{x_i \to 0} T_i(x_i) = t_i \ \& \ \lim_{x_i \to c_i} T_i(x_i) = +\infty$$

όπου:

- t_i [min]: ο σταθερός χρόνος που απαιτείται για να κινηθούμε στο δρόμο i όταν η κίνηση είναι ασθενής ($x_i \to 0$)
- c_i [οχ./min]: ο μέγιστος δυνατός ρυθμός διέλευσης οχημάτων από τον δρόμο i (**Περιορισμός**: $x_i < c_i$)
- a_i συντελεστής για τον οποίο έχουμε:
  - Για $i = 1, \dots, 5$, ισχύει $a_i = 1.25$
  - Για $i = 6, \dots, 10$, ισχύει $a_i = 1.5$
  - Για $i = 11, \dots, 17$, ισχύει $a_i = 1$

Η τιμές του c_i για κάθε δρόμο (για κάθε ακμή στον γράφο) αναγράφονται με κόκκινο χρώμα στο Figure 1, ενώ ο ρυθμός/ροή εισερχόμενων οχημάτων V θεωρείται - για την αρχική μας ανάλυση - σταθερός και ίσος με 100 [οχ./min].

Στην συνέχεια παρατίθεται η μαθηματική διατύπωση του προβλήματος και η ανάλυση του γενετικού αλγορίθμου (GA) που θα το λύσει (Θέμα 1). Έπειτα ακολουθεί μια πιο λεπτομερής τεχνική παρουσίαση του GA μαζί με τα αποτελέσματα αυτού (Θέμα 2). Τέλος, κλείνουμε με την επίλυση του προβλήματος (με την ίδια μεθοδολογία) για ρυθμό εισερχομένων οχημάτων V που μπορεί να μεταβάλλεται μέχρι $\pm 15\%$ της αρχικής του τιμής (Θέμα 3).

Θέμα 1

Συνάρτηση Προς Ελαχιστοποίηση

Για να διατυπώσουμε μαθηματικά το πρόβλημα θα πρέπει πρώτα να ορίσουμε μία συνάρτηση προς ελαχιστοποίηση. Όπως αναφέραμε και στην εισαγωγή, μας ενδιαφέρει η ελαχιστοποίηση ως προς x_i του συνολικού χρόνου διάσχισης του δικτύου ανά όχημα. Για να το πετύχουμε αυτό επιλέγουμε να ελαχιστοποιήσουμε το ***άθροισμα*** των χρόνων διέλευσης οχημάτων από ***όλες*** τις ακμές του δικτύου, ορίζοντας για **συνάρτηση προς ελαχιστοποίηση** την:

$$S(X) = T_1(x_1) + T_2(x_2) + \cdots + T_{17}(x_{17}) = \sum_{i=1}^{17} T_i(x_i), \quad \text{όπου } T_i(x_i) = t_i + \frac{a_i x_i}{1 - \frac{x_i}{c_i}},$$

με την $S(X)$ να είναι συνάρτηση **όλων** των ρυθμών/ροών x_i, όπου $X = [x_1\ x_2\ \ldots\ x_{17}]$.

Στην παραπάνω συνάρτηση που ορίσαμε παρατηρούμε πως η τιμή του t_i, η οποία αντιπροσωπεύει τον σταθερό χρόνο για να κινηθούμε στον δρόμο με ασθενή κίνηση, *δεν επηρεάζει* τη διαδικασία ελαχιστοποίησης της συνάρτησης. Αυτό συμβαίνει επειδή στην $S(X)$ οι όροι αυτοί απλώς προστίθενται προκαλώντας ένα σταθερό προσθετικό "κόστος" (*overhead*) που δεν μπορούμε να το μεταβάλλουμε με τα x_i. Οπότε στην ανάλυσή μας - χωρίς βλάβη της γενικότητας του αποτελέσματος της βελτιστοποίησης και για απλότητα στο υπολογιστικό κόστος - ***μπορούμε να θεωρήσουμε όλα τα*** t_i ***ίσα με μηδέν***.

Περιορισμοί

Έχοντας ορίσει την συνάρτηση προς ελαχιστοποίηση θα παραθέσουμε τώρα όλους τους περιορισμούς που τα x_i πρέπει να τηρούν ώστε η λύση να είναι αποδεικτή:

- Όπως ήδη αναφέραμε από την εισαγωγή πρέπει ο ρυθμός οχημάτων που ***εισέρχονται*** σε κάθε κόμβο να **ισούται** με τον ρυθμό που αυτά ***εξέρχονται*** από αυτόν τον κόμβο. Συνολικά στον γράφο μας υπάρχουν 9 κόμβοι επομένως θα πρέπει να τηρούνται συνολικά οι 9 εξισώσεις:
  1. $x_1 + x_2 + x_3 + x_4 = V$
  2. $x_1 = x_5 + x_6$
  3. $x_2 = x_7 + x_8$
  4. $x_4 = x_9 + x_{10}$
  5. $x_3 + x_8 + x_9 = x_{11} + x_{12} + x_{13}$
  6. $x_6 + x_7 + x_{13} = x_{14} + x_{15}$
  7. $x_5 + x_{14} = x_{16}$
  8. $x_{10} + x_{11} = x_{17}$
  9. $x_{17} + x_{12} + x_{15} + x_{16} = V$
- Επίσης είδαμε ήδη πως κάθε ακμή έχει ένα *μέγιστο επιτρεπτό ρυθμό διέλευσης οχημάτων* τον οποίο δεν μπορεί να ξεπεράσει, αλλά ούτε και να φτάσει διότι τότε θα είχαμε για αυτόν τον δρόμο $T_i = \infty$. Επομένως, πρέπει $x_i < c_i,\ \forall i \in \{1, \ldots, 17\}$.
- Τέλος μένει να αναφέρουμε τον περιορισμό $x_i \geq 0,\ \forall i \in \{1, \ldots, 17\}$, μιας και δεν επιτρέπεται να έχουμε αρνητική ροή κυκλοφορίας/διέλευσης οχημάτων.

Σχόλιο*: Για το Θέμα 1 έχουμε* ***σταθερό*** *V και ίσο με 100.*

Γενετικός Αλγόριθμος

Εισαγωγή

Έχοντας διατυπώσει μαθηματικά το πρόβλημα βελτιστοποίησης που καλούμαστε να επιλύσουμε, είμαστε πλέον σε θέση να διατυπώσουμε με ακρίβεια και τον μαθηματικό μηχανισμό μέσω του οποίου ο γενετικός αλγόριθμος θα προσφέρει τη λύση.

Κάθε γενετικός αλγόριθμος (GA) λειτουργεί προσομοιώνοντας τη διαδικασία της φυσικής εξέλιξης, με στόχο την εύρεση βέλτιστων λύσεων σε προβλήματα, όπως το δικό μας, με πολλαπλές παραμέτρους. Συνοπτικά η διαδικασία που ακολουθείται ξεκινάει με τη δημιουργία ενός ***αρχικού πληθυσμού*** πιθανών λύσεων (που ικανοποιούν τις προδιάγραφες-περιορισμούς), με κάθε λύση να αναπαρίσταται ως ένα χρωμόσωμα.

Χρωμόσωμα (Chromosome)

Το χρωμόσωμα (chromosome) ορίζεται ως ένα διάνυσμα παραμέτρων/γονιδίων (genes) που μπορεί να κωδικοποιηθεί με διάφορους τρόπους (δυαδικά, με πραγματικές τιμές, κ.α.). Δεδομένου ότι στο δικό μας πρόβλημα μπορούμε να μεταβάλουμε μόνο τα x_i, τα οποία είναι floating point αριθμοί, θα ακολουθήσουμε την κωδικοποίηση με πραγματικές τιμές. Άρα κάθε χρωμόσωμα θα αναπαρίσταται ως ένας πίνακας γραμμής της μορφής $X = [x_1\ x_2\ \dots\ x_{17}]$ με όλα τα x_i να πρέπει να ικανοποιούν τους προαναφερθείς περιορισμούς.

Ο αρχικός πληθυσμός περνάει από την φάση της ***εξέλιξης (evolution)*** η οποία πραγματοποιείται μέσω των διαδικασιών επιλογής (selection), διασταύρωσης (crossover) και μετάλλαξης (mutation) των χρωμοσωμάτων του πληθυσμού. Έπειτα, για κάθε χρωμόσωμα του *νέου πληθυσμού*, αξιολογείται η "ποιότητα" κάθε λύσης/χρωμοσώματος μέσω μιας συνάρτησης καταλληλότητας, fitness function, η οποία καθορίζει το πόσο καλά προσαρμόζεται η εκάστοτε λύση στο υπό εξέταση πρόβλημα.

Fitness Function

Έστω $f(X)$ η συνάρτηση καταλληλότητας για τα χρωμοσώματα $X = [x_1\ x_2\ \dots\ x_{17}]$ του προβλήματός μας. Τότε για να εκφράζει η συνάρτηση αυτή το πόσο καλά προσαρμόζεται η εκάστοτε λύση στο υπό εξέταση πρόβλημα θα πρέπει να επιστρέφει μεγαλύτερες τιμές για τις λύσεις/χρωμοσώματα του πληθυσμού τα οποία δίνουν μικρότερους συνολικούς χρόνους διέλευσης $S(X)$.

Μια συνάρτηση που επιστρέφει βαθμολογία για κάθε χρωμόσωμα σύμφωνα με τα παραπάνω θα μπορούσε να είναι η $f_1(X) = 1/S(X)$. Όντως, η $f_1(X)$ αποτελεί μια έγκυρη fitness function που επιστρέφει μεγαλύτερες βαθμολογίες για λύσεις που δίνουν μικρότερο $S(X)$. Όμως, το βασικό ελάττωμα μίας τέτοιας συνάρτησης είναι ότι σε περίπτωση που $S(X) \gg 1$, η $f_1(X)$ δεν θα είναι ιδιαίτερα ευαίσθητη σε μικρές αλλά ουσιώδεις μεταβολές του $S(X)$. Για παράδειγμα, έστω για δύο χρωμοσώματα X_1 & X_2 ότι $S(X_1) = S$ και $S(X_2) = S - \epsilon$, τότε για $S \gg \epsilon$ θα έχουμε $f_1(X_1) \simeq f_1(X_2)$.
Για να αυξήσουμε την ευαισθησία της $f_1(X)$, αν γνωρίζουμε ότι $S(X) > \omega$ για κάθε επιτρεπτό χρωμόσωμα X, τότε μπορούμε να λύσουμε το παραπάνω πρόβλημα επιλέγοντας $f_1^{'}(X) = \frac{1}{S(X)-\omega}$ ή για ακόμα μεγαλύτερη ευαισθησία $f_1^{''}(X) = \frac{1}{(S(X)-\omega)^k}$, $k \in \{2, 3, \dots\}$.

Εμείς, για την υλοποίησή μας στο MALAB - εκτός της $f_1^{''}(X)$ για $k = 6$ και $\omega = 700$ ($S(X) > 700$ από *trial and error* για $V = 100$) - θα εισάγουμε ως fitness function και την:

$$f_2(X) = e^{-\alpha \cdot S(X)} \quad ,$$

όπου α ένας συντελεστής που δηλώνει την ευαισθησία που θα έχει η $f_2(X)$ στις μεταβολές του $S(X)$.
Ετσί, ακόμα και για μιρκή μεταβολή ϵ στον συνολική χρόνο S ($S \gg \epsilon$), θα έχουμε:

$$\frac{f_2(X_2)}{f_2(X_1)} = \frac{e^{-\alpha \cdot (S-\epsilon)}}{e^{-\alpha \cdot S}} = e^{\alpha \cdot \epsilon}$$

Έπεται ότι ρυθμίζοντας κατάλληλα το α θα μπορούμε να έχουμε την επιθυμητή ευαισθησία. Επιλέγοντας για παράδειγμα $\alpha = 0.05$, θα έχουμε περίπου διπλασιασμό της βαθμολογίας f_2 όταν $\epsilon = 15$ $[min]$, ανεξαρτήτως του πόσο μεγάλο ήταν το αρχικό S, ενώ για μεγαλύτερες μεταβολές ϵ θα κυριαρχεί ο μικρότερος χρόνος.

Η διαδικασία αυτή επαναλαμβάνεται με τον νέο πληθυσμό να περνάει εκ νέου απο την φάση της εξέλιξης (selection-crossover-mutation), έως ότου επιτευχθεί η επιθυμητή ακρίβεια της λύσης˙ δηλαδή όταν ικανοποιηθεί η συνθήκη τερματισμού.

Συνθήκη Τερματισμού

Όπως με την fitness function είχαμε πολλές εναλλακτικές που θα μπορούσαμε να ορίσουμε έτσι κι εδώ υπάρχουν αρκετές μεθοδολογίες τερματισμού στην βιβλιογραφία των GA.
Στον δικό μας αλγόριθμο, ο τερματισμός επιτυγχάνεται με δύο βασικά κριτήρια. Ξεκινάμε με το να εκτελούμε έναν προκαθορισμένο αριθμό επαναλήψεων (`maxGenerations`) και τερματίζουμε μόλις ολοκληρωθεί αυτός ο αριθμός. Ταυτόχρονα, εφαρμόζουμε έναν μηχανισμό πρόωρου τερματισμού, ελέγχοντας τη "σταθερότητα" των βέλτιστων χρόνων ανά γενιά. Συγκεκριμένα, εάν για τις **τελευταίες** n **γενιές** η *τυπική απόκλιση* των βέλτιστων χρόνων είναι μικρότερη από ένα προκαθορισμένο όριο (`tolerance`), τότε ο αλγόριθμος τερματίζει πρόωρα. Αυτό σημαίνει ότι η τιμή της λύσης έχει σταθεροποιηθεί και η περαιτέρω εκτέλεση του αλγορίθμου δεν αναμένεται να προσφέρει σημαντική βελτίωση.
Σχόλιο*: Βέλτιστος χρόνος μιας γενιάς είναι η καλύτερη/μικρότερη τιμή $S(X)$ που εμφανίζεται στον πληθυσμό της.*

Selection

Εμβαθύνοντας στην διαδικασία της εξέλιξης του πληθυσμού ήρθε η ώρα να μιλήσουμε για την πρώτη φάση, αυτήν του selection.

Μετά τη δημιουργία μίας γενιάς, η fitness function αξιολογεί τις λύσεις, αποδίδοντας μια βαθμολογία σε κάθε χρωμόσωμα, η οποία ανταναικλά την ποιότητα της λύσης. Όπως ήδη αναλύσαμε εκτενώς, όσο υψηλότερη είναι η τιμή της βαθμολογίας αυτής, τόσο πιο κοντά βρίσκεται η λύση στη βέλτιστη. Ωστόσο, δεδομένου ότι το πρόβλημα περιλαμβάνει πολλαπλές παραμέτρους, η απόρριψη των λιγότερο αποτελεσματικών λύσεων δεν πρέπει να γίνεται αλόγιστα, καθώς μπορεί να περιέχουν στοιχεία που θα συμβάλουν στη βελτίωση της λύσης στις επόμενες γενιές. Για να διατηρηθεί η ποικιλομορφία του πληθυσμού και να αποφευχθεί η πρόωρη σύγκλιση σε υπο-βέλτιστες λύσεις (τοπικά βέλτιστα), η επιλογή των χρωμοσωμάτων που θα επιβιώσουν πραγματοποιείται με έναν πιθανοκρατικό μηχανισμό, γνωστό ως επιλογή μέσω ρουλέτας (roulette wheel selection).

Η διαδικασία της ρουλέτας ξεκινά με την κανονικοποίηση όλων των fitness scores των χρωμοσωμάτων του πληθυσμού και έπειτα με τον υπολογισμό της *αθροιστικής πιθανότητας επιλογής*, για κάθε ένα από αυτά. Με αυτόν τον τρόπο, κατασκευάζεται νοητά ένα κυκλικό διάγραμμα (ρουλέτα), όπου το μέγεθος κάθε τμήματος είναι ανάλογο της *κανονικοποιημένης* τιμής του fitness score του χρωμοσώματος που αντιπροσωπεύει. Έτσι, με την τυχαία παραγωγή αριθμών στο διάστημα $[0, 1]$, επιλέγονται τα χρωμοσώματα που θα επιβιώσουν και θα βρίσκονται και στην επόμενη γενιά.

Όπως είναι φανερό, οι καλύτερες λύσεις καταλαμβάνουν μεγαλύτερο μέρος του διαγράμματος και έχουν αυξημένες πιθανότητες να επιλεγούν, ενώ οι λιγότερο αποδοτικές λύσεις έχουν μικρότερη πιθανότητα επιβίωσης. Η υλοποίηση της συνάρτησης επιλογής στο MATLAB βρίσκεται στο αρχείο `src/rouletteWheelSelection.m`

Crossover

Αφότου επιλέξουμε ποια χρωμοσώματα του πληθυσμού θα επιβιώσουν περνάμε στην φάση του crossover.

Κι εδώ έχουμε αρκετές εναλλακτικές τις οποίες μπορούμε να υλοποιήσουμε. Λαμβάνοντας όμως υπόψιν το πως έχουμε κωδικοποιήσει το χρωμόσωμα και ότι οι περιορισμοί που έχουμε για τις μεταβλητές x_i (δηλαδή για τα γονίδια του χρωμοσώματος) είναι αυστηροί (πολλοί εκ των οποίων είναι ισότητες που πρέπει να τηρούνται) επιλέγουμε μέθοδο διασταύρωσης που βασίζεται στον **υπολογισμό του μέσου όρου**. Έτσι, ο αλγόριθμος για την παραγωγή κάθε απογόνου, επιλέγει τυχαία δύο χρωμοσώματα και δημιουργεί ένα νέο, όπου κάθε στοιχείο του x_i προκύπτει ως ο αριθμητικός μέσος των αντίστοιχων στοιχείων των γονικών χρωμοσωμάτων:

$$x_{offspring,i} = \frac{x_{parent1,i} + x_{parent2,i}}{2}$$

Αυτή η προσέγγιση επιτρέπει μία ομαλή προσαρμογή των λύσεων, αποφεύγοντας απότομες αλλαγές που θα μπορούσαν να οδηγήσουν σε μη έγκυρες λύσεις. Είναι σημαντικό να σημειωθεί ότι το crossover δεν εφαρμόζεται σε όλα τα χρωμοσώματα του πληθυσμού, καθώς έχει εισαχθεί μια πιθανότητα 20% να επιστραφεί ένας γονέας αντί του offspring. Επιπλέον, σε περιπτώσεις όπου ο απόγονος που προκύπτει δεν ικανοποιεί τους περιορισμούς του προβλήματος, ο αλγόριθμος δεν τον συμπεριλαμβάνει στον νέο πληθυσμό. Αντί αυτού, διατηρεί και πάλι έναν από τους δύο γονείς, εξασφαλίζοντας ότι όλες οι λύσεις της επόμενης γενιάς παραμένουν έγκυρες. Μετά τις διασταυρώσεις ο νέος πληθυσμός έχουμε εξασφαλίσει να έχει το ίδιο μέγεθος με τον προ crossover.

Για τεχνικές λεπτομέρειες για την υλοποίηση της συνάρτησης μπορείτε να ανατρέξτε στον κώδικα του αρχείου: `src/crossover.m`

Mutation

Η διαδικασία της μετάλλαξης εφαρμόζεται στα χρωμοσώματα του πληθυσμού μετά τη διασταύρωση, με σκοπό την εισαγωγή μικρών τυχαίων παραλλαγών που ενισχύουν την εξερεύνηση του χώρου λύσεων. Ορίσαμε μια πιθανότητα μετάλλαξης (`mutationRate`) ίση με 25%, που σημαίνει ότι για κάθε χρωμόσωμα υπάρχει 25% πιθανότητα να υποβληθεί σε μεταβολές. Η μετάλλαξη πραγματοποιείται με τη βοήθεια μικρού θορύβου, ο οποίος παράγεται από μια κανονική κατανομή με τυπική απόκλιση σ (`sigma`) ίση με 0.5 και προστίθεται σε όλα τα γονίδια του χρωμοσώματος με τρόπο τέτοιο ώστε το τελικό αποτέλεσμα να συνεχίζει να είναι έγκυρο. Σε κάθε περίπτωση, κατά την ολοκλήρωση των παραπάνω μεταβολών, η νέα διαμορφωμένη λύση επαληθεύεται για να ελεγχθεί ότι όντως πληροί όλους τους περιορισμούς του προβλήματος. Με αυτόν τον τρόπο, η διαδικασία της μετάλλαξης διασφαλίζει ότι οι λύσεις παραμένουν έγκυρες, ενώ ταυτόχρονα εισάγουν μικρές, ελεγχόμενες αλλαγές (θόρυβο) που μπορούν να οδηγήσουν σε περαιτέρω βελτιστοποίηση εξερευνώντας ολόκληρο τον χώρο των δυνατών λύσεων.

Η υλοποίηση του mutation παρατίθεται στο: `src/mutation.m`

Θέμα 2

Έχοντας ήδη διατυπώσει μαθηματικά τόσο το πρόβλημα όσο και τον μηχανισμό με τον οποίο ο γενετικός αλγόριθμος καταλήγει στην βέλτιστη λύση, μπορούμε τώρα να αναλύσουμε τα αποτελέσματα της υλοποίησής μας εξηγώντας παράλληλα σε μεγαλύτερο βάθος κάποια τεχνικά σημεία της υλοποίησης.

Το αρχείο `mainConstV.m` *περιέχει τον εκτελέσιμο κώδικα σε MATLAB που τρέχει τον γενετικό αλγόριθμο για σταθερό* $V(= 100)$.

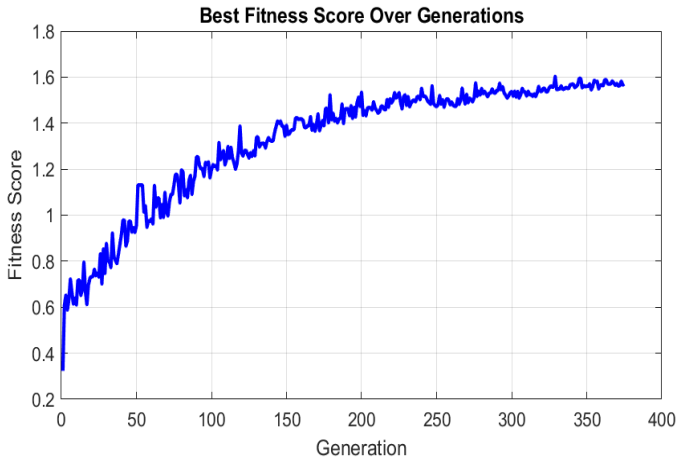
Ο αριθμός των γονιδίων του χρωμοσώματος (`chromosomeSize`) είναι 17 (από x_1 έως x_{17}) ενώ έχουμε επιλέξει ο αρχικός πληθυσμός να περιλαμβάνει συνολικά 500 τυχαία χρωμοσώματα (`populationSize`). Με εξαίρεση τον αρχικό πληθυσμό όμως, στο **Evolution Loop** έχουμε σταθερό πληθυσμό ίσο με 300 (`numOfSelections`). Δηλαδή η συνάρτηση επιλογής (selection) διαλέγει τυχαία (σύμφωνα με τον κανόνα που αναλύσαμε) 300 χρωμοσώματα που θα επιβιώσουν στην επόμενη γενιά. Έπειτα, η κάθε γενιά έχει πάντα 300 χρωμοσώματα στον πληθυσμό της με την συνάρτηση επιλογής αντί να ελαττώνει σε κάθε επανάληψη τον πληθυσμό, να κάνει "τυχαία" selections που να οδηγούν και πάλι σε συνολικό πληθυσμό 300. Μιας και υπάρχει δυνατότητα επανεπιλογής χρωμοσώματος στην φάση του selection, να αναφέρουμε πως εν τέλη - μετά το crossover και το mutation - έχουμε (όπως φαίνεται και από τα *log output*) επαρκή αριθμό **διακριτών** χρωμοσωμάτων προς αξιολόγηση από την fitness function.

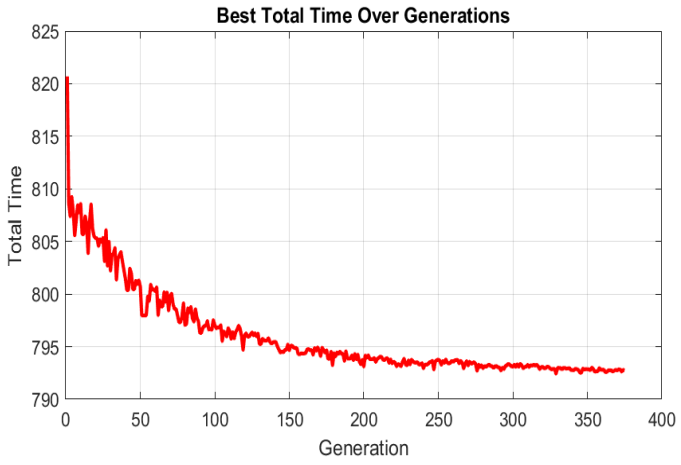
Επίσης, πριν τρέξουμε τον αλγόριθμο, να πούμε πως οι συναρτήσεις `fitnessScoreFunc` και `inverseFitnessScoreFunc` αναφέρονται στην συνάρτηση καταλληλότητας που θα χρησιμοποιηθεί και στην αντίστροφη συνάρτηση από την οποία προκύπτει ο συνολικός χρόνος $S(X)$ στον οποίο αντιστοιχείται το δοθέν σκορ. Όπως ήδη εξηγήσαμε, έχουμε υλοποιήσει δύο διαφορετικές fitness function με τις οποίες σκοπεύουμε να τρέξουμε τον κώδικα για να δούμε και πως αυτές επηρεάζουν την ταχύτητα σύγκλισης του αλγορίθμου.

Τέλος, για την συνθήκη τερματισμού δίνουμε `tolerance=0.1` και παράθυρο `n=20` (για την θεωρητική ανάλυση της συνθήκης τερματισμού αναφερθήκαμε στο αντίστοιχο section του *Θέματος 1*). Αυτές οι απαιτήσεις είναι ιδιαίτερα αυστηρές μιας και δείχνουν πως για να σταματήσουμε τον αλγόριθμο σε γενιά μικρότερη της `maxGenerations=1200` περιμένουμε την τυπική απόκλιση των βέλτιστων λύσεων των τελευταίων 20 γενεών να γίνει μικρότερη του 0.1. Σε περίπτωση που επιθυμούμε **μεγαλύτερη ακρίβεια** μπορούμε να *αυξήσουμε* το μέγεθος του παραθύρου n από 20 σε 100 ή 200 ώστε να έχουμε μεγαλύτερη ακρίβεια στο τελικό αποτέλεσμα.

1. Αποτελέσματα για $f(X) = \frac{1}{(S(X)-700)^6}$

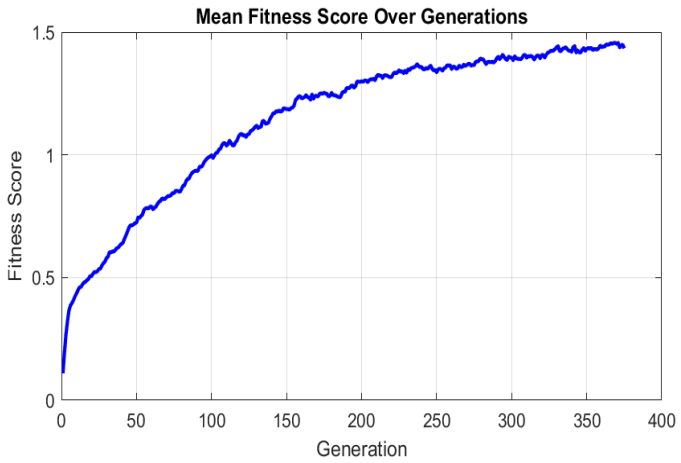
Τρέχοντας τον κώδικα παίρνουμε τα παρακάτω διαγράμματα:

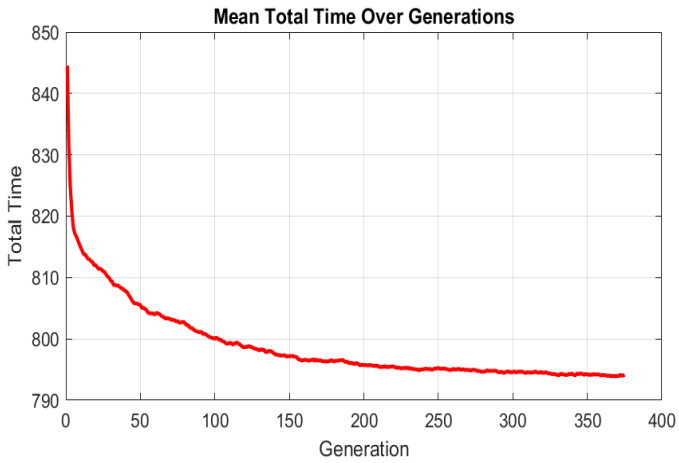




Το πρώτο διάγραμμα δείχνει την τιμή του *καλύτερου* ***fitness score*** που εμφανίζεται στον πληθυσμό ανά γενιά, ενώ το δεύτερο την τιμή του *καλύτερου* ***συνολικού χρόνου*** ανά γενιά. Και από τις δύο γραφικές παραστάσεις είναι εμφανές ότι ο γενετικός αλγόριθμος συγκλίνει μετά την γενιά 300, τερματίζοντας στην 375. Ο λόγος για τον οποίο εμφανίζεται θόρυβος (διακυμάνσεις) στα αποτελέσματα ανά γενιά είναι κυρίως λόγο των mutations (αλλά και των crossover) που έχουν σαν αποτέλεσμα να επηρεάζουν - με μικρές "κινήσεις" όπως τις αναλύσαμε στο *Θέμα 1* - τον καλύτερο χρόνο που εμφανίζεται στον πληθυσμό κάθε γενιάς.

Έχοντας επίσης αποθηκεύσει για κάθε γενιά το μέσο fitness score όλου του πληθυσμού (άρα μπορούμε να υπολογίσουμε και τον μέσο χρόνο) παραθέτουμε και τα αντίστοιχα διαγράμματα που αυτήν την φορά αναφέρονται στις μέσες τιμές του fitness score f και total time S ανά γενιά (για όλον τον πληθυσμό):





Και εδώ παρατηρούμε πως υπάρχει θόρυβος, αλλά αυτήν την φόρα είναι ασθενέστερος, μιας και εφόσον παίρνουμε τις μέσες τιμές ανά γενιά για όλον τον πληθυσμό, βγάζουμε ένα πιο σταθερό αποτέλεσμα που δεν εξαρτάται τόσο έντονα από μόνο ένα χρωμόσωμα όπως στην περίπτωση του βέλτιστου (best) χρόνου και score.

Παρακάτω παραθέτουμε τα ακριβή τελικά αποτελέσματα (best fitness, best total time και best chromosome/solution $[x_1\ x_2\ \ldots\ x_{17}]$) που προέκυψαν από την εκτέλεση του αλγορίθμου.

```
Converged at generation 375 with best fitness: 1.571282e-12, total time: 792.745101
Optimization finished. Best solution found:
  Columns 1 through 11

   31.5765   14.0205   23.5018   30.9012   20.1781   11.3983   10.7057    3.3148   14.4794   16.4218   10.7681

  Columns 12 through 17

   19.0122   11.5158    6.7313   26.8885   26.9094   27.1899

Resulted Chromosome Valid !!
```

Σχήμα 4: Αποτελέσμα GA για V = 100

Σχόλια:

1. Μιας και πολλά τμήματα του γενετικού αλγορίθμου λειτουργούν ***πιθανοκρατικά*** τα παραπάνω αποτελέσματα που δόθηκαν είναι *ενδεικτικά*. Τρέχοντας τον αλγόριθμο αρκετές φορές βλέπουμε πως έχουμε παραπλήσια αποτελέσματα σε κάθε περίπτωση, με το συνολικό πλήθος το γενεών να κυμαίνεται από 150 έως 600.
2. Στο εκτελέσιμο αρχείο υπάρχει η Boolean μεταβλητή `printLogs` την οποία αν θέσουμε *true*, θα έχουμε για κάθε γενιά πληροφορίες σχετικά με το πόσα unique χρωμοσώματα επιλέχτηκαν, πόσα crossover και mutations έγιναν και είναι αποδεκτά, και πόσα διακριτά χρωμοσώματα έχει (μετά τις αλλαγές) ο νέος πληθυσμός.
3. Στο αρχείο `main_fmincon.m` χρησιμοποιούμε την **fmincon** συνάρτηση του MATLAB για τον υπολογισμό του ***ακριβούς βέλτιστου/ελάχιστου χρόνου***. Αυτό το αρχείο είναι ανεξάρτητο από τα υπόλοιπα, επομένως όσες συναρτήσεις απαιτούνται για να τρέξει η fmincon υλοποιούνται μέσα σε αυτό το αρχείο. Τρέχοντάς το για $V = 100$ παίρνουμε ως βέλτιστη λύση την: Figure 5. Όπως βλέπουμε η δική μας λύση έχει απόκλιση από τον πραγματικό βέλτιστο χρόνο μόλις $\sim 0.1\%$. Ένας εύκολος τρόπος να αυξήσουμε ακομά περισσότερο την ευαισθησία του γενετικού μας αλγορίθμου είναι να μεγαλώσουμε το παράθυρο n από 20 σε 200.

```
Optimal Traffic Flow:
  Columns 1 through 11

   31.7086   14.0588   23.2000   31.0326   20.1164   11.5922   11.0675    2.9913   14.3303   16.7023   10.5689

  Columns 12 through 17

   19.4967   10.4560    5.3454   27.7703   25.4618   27.2712

Minimum Total Travel Time: 791.964710 min
```

Σχήμα 5: Αποτελέσμα fmincon για V = 100

2. Αποτελέσματα για $f(X) = e^{-\alpha \cdot S(X)}$

Κάνοντας comment out τις συναρτήσεις:

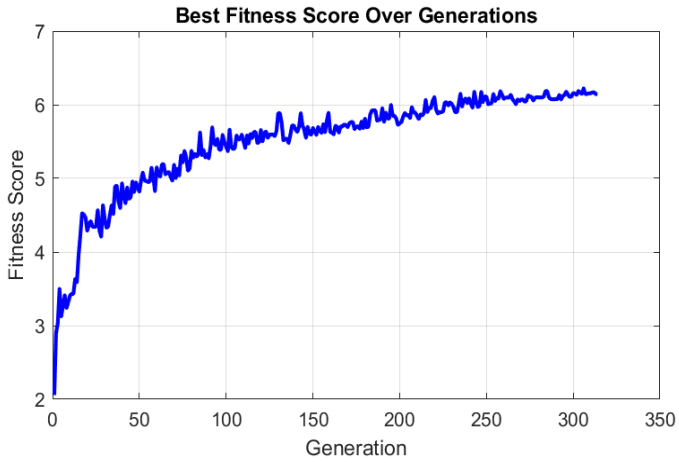
```
fitnessScoreFunc        = @(totalTime) 1 / (totalTime - 700)^6;
inverseFitnessScoreFunc = @(fitnessScore) 1 / fitnessScore^(1/6) + 700;
```

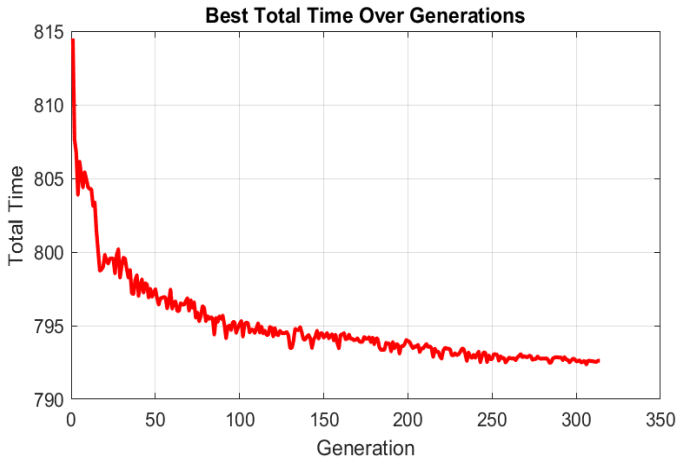
Και comment in τις:

```
alpha = 0.05;
fitnessScoreFunc        = @(totalTime) exp(-alpha * (totalTime));
inverseFitnessScoreFunc = @(fitnessScore) - log(fitnessScore)/alpha;
```

έχουμε πλέον αλλάξει την συνάρτηση $f(X)$ που θα χρησιμοποιεί ο γενετικός μας αλγόριθμος σε όλη την υλοποίηση. Σε αντίθεση με την προηγούμενη περίπτωση που μέσω trial and error έπρεπε να βγάλουμε ότι $S(X) > 700$, με αυτήν την συνάρτηση fitness δεν χρειάζεται να δώσουμε *καμία* τέτοια πληροφορία στον κώδικα. Η μόνη σταθερά που αξιοποιεί ο τύπος είναι η α η οποία εκφράζει απλώς την ευαισθησία που θέλουμε να δώσουμε.

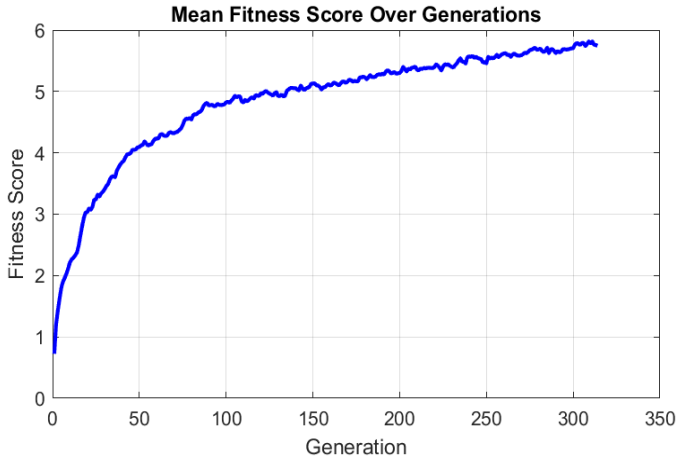
Τρέχοντας και πάλι τον κώδικα, παίρνουμε τα παρακάτω διαγράμματα:

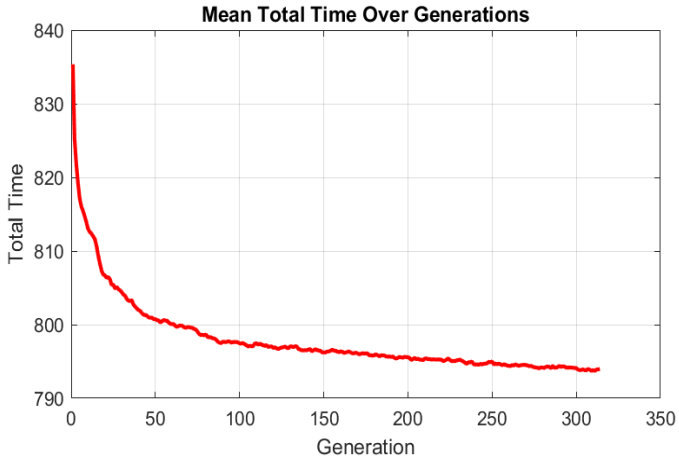




Το πρώτο διάγραμμα δείχνει την τιμή του *καλύτερου* ***fitness score*** που εμφανίζεται στον πληθυσμό ανά γενιά (σε κλίμακα 10^{-18}), ενώ το δεύτερο την τιμή του *καλύτερου* ***συνολικού χρόνου*** ανά γενιά (σε minutes). Και από τις δύο γραφικές παραστάσεις είναι εμφανές ότι ο γενετικός αλγόριθμος συγκλίνει και πάλι μετά την γενιά 300, τερματίζοντας στην 314. Ο λόγος για τον οποίο εμφανίζεται θόρυβος (διακυμάνσεις) στα αποτελέσματα είναι ο ίδιος με την προηγούμενη περίπτωση.

Έχοντας επίσης αποθηκεύσει για κάθε γενιά το μέσο fitness score όλου του πληθυσμού (άρα μπορούμε να υπολογίσουμε και τον μέσο χρόνο) παραθέτουμε και τα αντίστοιχα διαγράμματα που αυτήν την φορά αναφέρονται στις μέσες τιμές:





Και εδώ παρατηρούμε πως υπάρχει θόρυβος, αλλά αυτήν την φόρα (όπως και με την προηγούμενη περίπτωση) είναι ασθενέστερος.

Παρακάτω παραθέτουμε τα ακριβή τελικά αποτελέσματα (best fitness, best total time και best chromosome/solution) που προέκυψαν από την εκτέλεση του αλγορίθμου για αυτό το fitness function:

```
Converged at generation 314 with best fitness: 6.144360e-18, total time: 792.619941
Optimization finished. Best solution found:
  Columns 1 through 11

   32.2474   14.0055   23.1999   30.5472   20.5154   11.7320   10.3948    3.6107   13.3845   17.1627   10.3074

  Columns 12 through 17

   19.3523   10.5354    4.7659   27.8962   25.2813   27.4702

Resulted Chromosome Valid !!
```

Σχήμα 8: Αποτελέσμα για V = 100 (με το εκθετικό f(X))

Σχόλια:

1. Και πάλι, τα αποτελέσματα που παρουσιάζουμε εδώ είναι ενδεικτικά. Τρέχοντας τον αλγόριθμο αρκετές φορές βλέπουμε πως έχουμε κάθε φορά παραπλήσια αποτελέσματα, με το συνολικό πλήθος το γενεών να κυμαίνεται συνήθως από 200 έως 500.
2. Η τάξη μεγέθους του fitness function είναι (για τον συγκεκριμένο τύπο) 10^{-18}, άρα και τα αντίστοιχα διαγράμματα είναι αυτής της κλίμακας. Το MATLAB μπορεί να χειρίζεται αρκετά μικρούς αριθμούς έχοντας ιδιαίτερα καλή ευαισθησία - ιδιαίτερα για τιμές κοντά στο μηδέν - οπότε δεν χρειάζεται να πολλαπλασιάσουμε χειροκίνητα το fitness score με κάποια σταθερά για να αλλάξουμε την τάξη μεγέθους.
3. Τρέχοντας τον κώδικα αρκετές φορές, δοκιμάζοντας και τις δύο fitness function, παρατηρούμε πως και οι δύο δίνουν κατά μέσο την ίδια ακρίβεια τελικού αποτελέσματος επιτρέποντας σύγκληση πολύ κοντά στο πραγματικό ολικό ελάχιστο. Όσον αφορά τον συνολικό αριθμό των γενεών που απαιτούνται, αυτός φαίνεται να μην διαφέρει σημαντικά και στις δύο περιπτώσεις (δεν μπορούμε να απορρίψουμε την μηδενική υπόθεση ότι ο αριθμός των γενεών είναι κατά μέσο όρο ίσος και στις δύο περιπτώσεις). Σε κάθε περίπτωση, το πλήθος των γενεών δεν εξαρτάται αμιγώς από την συνάρτηση καταλληλότητας αλλά σημαντικό ρόλο παίζουν τόσο ο αρχικός πληθυσμός όσο και τα τυχαία mutation και crossover (αλλά και selection μιας και έχουμε υλοποιήσει και εκεί πιθανοκρατικό μηχανισμό επιλογής).

Γενική Σημείωση:
Ο αλγόριθμός μας τρέχει για μεγάλο αριθμό γενεών με άνω όριο τις $maxGenerations = 1200$. Ο λόγος για τον οποίο δώσαμε ένα τέτοιο μεγάλο όριο είναι ακριβώς επειδή βασική μέριμνα κατά την ανάπτυξη του αλγορίθμου (και του κώδικα) ήταν η όσο το δυνατόν μεγαλύτερη ακρίβεια του τελικού μας αποτελέσματος. Σε περίπτωση που μας ενδιαφέρει περισσότερο η ταχύτερη εύρεση μίας λύσης τότε μπορούμε να ελαττώσουμε το *tolerance* που δίνουμε ή να μειώσουμε το μέγεθος του παραθύρου n. Θα μπορούσαν να πραγματοποιηθούν και άλλες αλλαγές (στο *population size*, στο σ (sigma) της κατανομής του θορύβου που προσθέτουμε, στην πιθανότητα να κάνουμε crossover και mutation, κ.α.) που θα οδηγούσαν επίσης σε μείωση του χρόνου εκτέλεσης του αλγορίθμου με κίνδυνο όμως ο αλγόριθμος τότε να σταματάει/τερματίζει σε υποβέλτιστες λύσεις.

Θέμα 3

Περνάμε τώρα στην επίλυση του *τρίτου Θέματος* στο οποίο ο ρυθμός V παύει πλέον να θεωρείται σταθερός και ίσος με 100 και μπορεί να μεταβάλλεται μέχρι $\pm 15\%$ της αρχικής του τιμής. Έχουμε επόμενος για το μεταβλητό V τον **περιορισμό** $85 \leq V \leq 115$.

Για την επίλυση του προβλήματος αυτού υπάρχουν δύο διαφορετικές προσεγγίσεις που θα ακολουθήσουμε τις οποίες παρουσιάζουμε στις παρακάτω υποενότητες.

1ος Τρόπος: Το V προστίθεται ως γονίδιο/παράμετρος στο χρωμόσωμα

Το αρχείο `mainRandV.m` *περιέχει τον εκτελέσιμο κώδικα σε MATLAB που τρέχει τον γενετικό αλγόριθμο, θεωρώντας το V ως ένα ακόμα (18ο) γονίδιο προς βελτιστοποίηση.*

Ο τρόπος που σχεδιάσαμε και υλοποιήσαμε ως τώρα τον αλγόριθμό μας στο MATLAB, μας επιτρέπει με ευκολία να ορίζουμε (αντί για 17 γονίδια/παραμέτρους ανά χρωμόσωμα) 18 παραμέτρους ανά χρωμόσωμα, με την τελευταία παράμετρο να αναφέρεται στο V. Άρα πλέον, όταν πάμε να ορίσουμε τυχαία τον αρχικό μας πληθυσμό, δίνουμε για κάθε χρωμόσωμα ξεχωριστά: **πρώτα** μια τυχαία τιμή στο V *μέσα στο επιτρεπτό εύρος* και **έπειτα** με βάση αυτήν ορίζουμε τις υπόλοιπες τυχαίες παραμέτρους, x_i, ώστε να ικανοποιούνται όλοι οι υπόλοιποι περιορισμοί. Έτσι ο *αρχικός πληθυσμός* έχει έγκυρα χρωμοσώματα, τα οποία έχουν τυχαίες/διάφορες τιμές V το καθένα.

Αυτό που περιμένουμε (λαμβάνοντας υπόψιν το πρόβλημα που καλούμαστε τώρα να λύσουμε) είναι εν τέλη, στον τελικό πληθυσμό, να έχουν επικρατήσει χρωμοσώματα με την μικρότερη τιμή V $(= 85)$ - μιας και μικρότερη ροή (άρα λιγότερη κίνηση) συνεπάγεται και μικρότερο συνολικό χρόνο $S(X)$.

Επίσης, όσον αφορά τις συναρτήσεις `fitnessScoreFunc` και `inverseFitnessScoreFunc`, θα συνεχίσουμε με την παρουσίαση μόνο της **εκθετικής**, καθώς όπως είδαμε η επιλογή της fitness function από τις δύο που παρουσιάσαμε δεν επηρεάζει αισθητά την τελική ακρίβεια του αποτελέσματος.

Σε περίπτωση που θέλουμε να χρησιμοποιήσουμε την εναλλακτική πρέπει να προσέξουμε να αλλάξουμε το κάτω όριο ω, ώστε $S(X) > \omega$, μιας και πλέον έχουμε και τιμές V που μπορεί να δίνουν $S(X) < 700$. Εμείς, για δοκιμή επιλέξαμε να δώσουμε τιμή $\omega = 500$ με το αποτέλεσμα να συγκλίνει στο βέλτιστο.

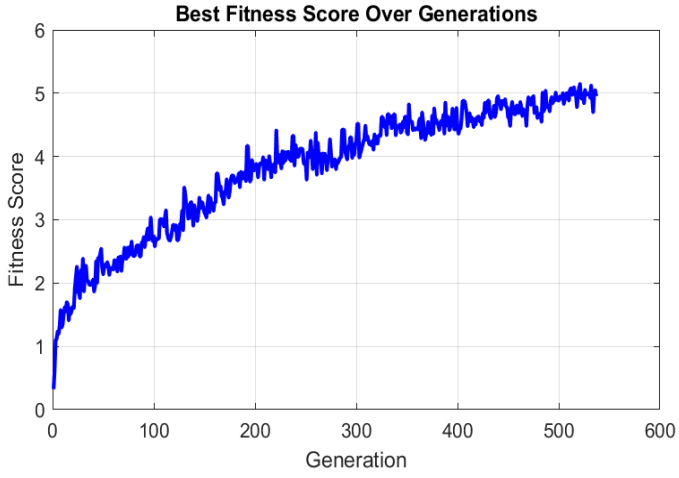
Τέλος, για την συνθήκη τερματισμού δίνουμε `tolerance=0.4` και παράθυρο `n=50`. Αυτές οι απαιτήσεις είναι και πάλι αυστηρές μιας και δείχνουν πως για να σταματήσουμε τον αλγόριθμο σε γενιά μικρότερη της `maxGenerations=1200` περιμένουμε την τυπική απόκλιση των βέλτιστων λύσεων των τελευταίων 50 γενεών να γίνει μικρότερη του 0.4.

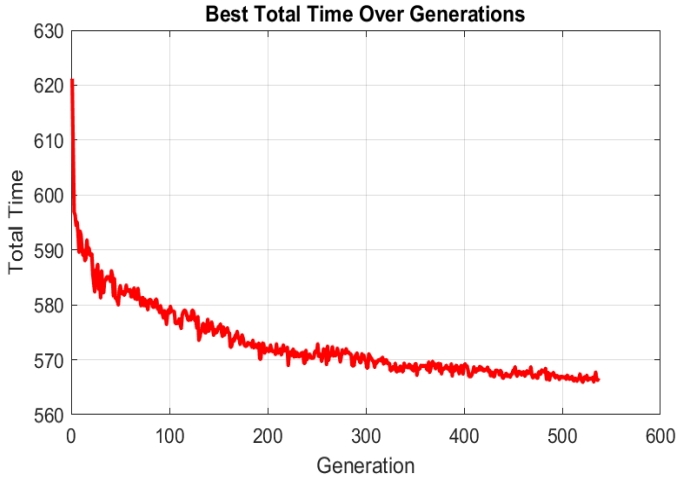
Ο λόγος για τον οποίο αυξήσαμε το tolerance είναι επειδή πλέον ακόμα και με μικρές αλλαγές (θόρυβο) στην παράμετρο V ο συνολικός χρόνος $S(X)$ μπορεί να αλλάζει αισθητά, επομένως, δίνοντας ένα μεγαλύτερο tolerance είμαστε πιο ανεκτικοί στις αλλαγές αυτές. Ταυτόχρονα, για να διατηρήσουμε την ακρίβεια σε ικανοποιητικά επίπεδα, αυξήσαμε την τιμή του παραθύρου n από 20 σε 50.

Σχόλια Υλοποίησης:

1. Δεδομένου ότι για όλα αρχικά V στα χρωμοσώματα του αρχικού πληθυσμού έχουμε $V \in [85, 115]$, όλα τα crossover θα δίνουν offspring με $V \in [85, 115]$ (καθώς $V_{offspring}$ είναι ο αριθμητικός μέσος των $V_{parent1}$ και $V_{parent2}$). Δεν επιτρέπουμε crossover μεταξύ χρωμοσωμάτων με μεγάλη διαφορά στην τιμή του V μιας και αν έχουμε $|V_{parent1} - V_{parent2}| > 1.5$ τότε μόνο ένα μικρό ποσοστό των τελικών offspring θα είναι valid $(< 50\%)$.
2. Στο mutation, πρώτα προσθέτουμε θόρυβο στην τιμή του γονιδίου/παραμέτρου V *προσέχοντας το αποτέλεσμα να είναι μέσα στα όρια* και μετά μεταβάλουμε όλες τις υπόλοιπες παραμέτρους (με τον τρόπο που αναφέραμε και στην προηγούμενη ανάλυση).
3. Ελέγχουμε πάντα τα αποτελέσματα ώστε να είναι valid πρωτού τα εισάγουμε στον νέο πληθυσμό.

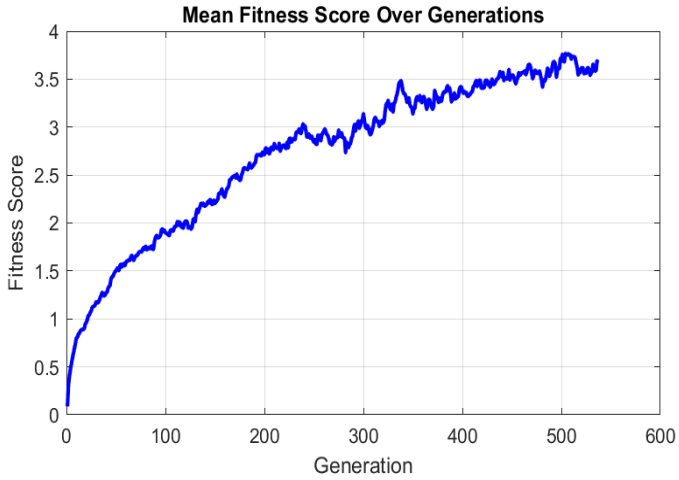
Τρέχοντας τον κώδικα παίρνουμε τα παρακάτω διαγράμματα:

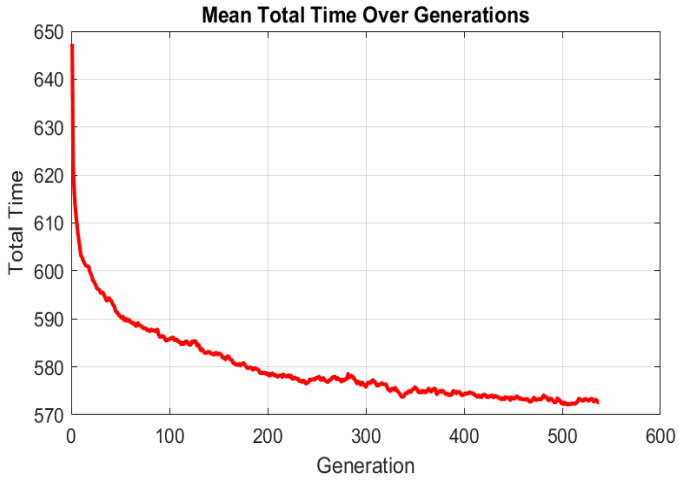




Το πρώτο διάγραμμα δείχνει την τιμή του *καλύτερου* ***fitness score*** που εμφανίζεται στον πληθυσμό ανά γενιά (είναι σε κλίμακα της τάξης $\times 10^{-13}$), ενώ το δεύτερο την τιμή του *καλύτερου* ***συνολικού χρόνου*** ανά γενιά. Και από τις δύο γραφικές παραστάσεις είναι εμφανές ότι ο γενετικός αλγόριθμος συγκλίνει μετά την γενιά 500, τερματίζοντας στην 537. Ο λόγος που εμφανίζεται περισσότερος θόρυβος (διακυμάνσεις) στα αποτελέσματα ανά γενιά, σε σχέση με την προηγούμενη ανάλυση, είναι λόγο των mutations που συμβαίνουν στο V που έχουν σαν αποτέλεσμα να επηρεάζουν, με μικρές "κινήσεις", ***όλες*** τις υπόλοιπες παραμέτρους x_i ώστε να ισχύον οι περιορισμοί.

Έχοντας επίσης αποθηκεύσει (όπως κάναμε και στο *Θέμα 2*) για κάθε γενιά το μέσο fitness score όλου του πληθυσμού (άρα μπορούμε να υπολογίσουμε και τον μέσο χρόνο) παραθέτουμε και τα αντίστοιχα διαγράμματα:





Και εδώ παρατηρούμε πως υπάρχει θόρυβος, αλλά αυτήν την φόρα είναι ασθενέστερος, μιας και εφόσον παίρνουμε τις μέσες τιμές ανά γενιά για όλον τον πληθυσμό, βγάζουμε ένα πιο σταθερό ανά γενία αποτέλεσμα.

Παρακάτω παραθέτουμε τα ακριβή τελικά αποτελέσματα (best fitness, best total time και best chromosome/solution $[x_1\ x_2\ \ldots\ x_{17}\ V]$) που προέκυψαν από την εκτέλεση του αλγορίθμου.

```
Converged at generation 537 with best fitness: 4.951980e-13, total time: 566.676376
Optimization finished. Best solution found:
  Columns 1 through 11

   26.9152   11.6174   20.6291   25.8530   17.2747    9.6405    9.2603    2.3571   13.1217   12.7314    8.0232

  Columns 12 through 18

   16.7213   11.3634    5.7772   24.4870   23.0519   20.7545   85.0147

Resulted Chromosome Valid !!
```

Σχήμα 11: Αποτελέσμα GA για μεταβλητό $V \in [85, 115]$

Σχόλια:

1. Τρέχοντας τον αλγόριθμο αρκετές φορές βλέπουμε πως έχουμε παραπλήσια αποτελέσματα σε κάθε περίπτωση, με το συνολικό πλήθος το γενεών να κυμαίνεται από 500 έως 900. Αν μας ενδιαφέρει ο αλγόριθμος να τερματίζει σε λιγότερες επαναλήψεις μπορούμε να αυξήσουμε το tolerance σε 1 ή να μειώσουμε το μέγεθος του παραθύρου n σε 20 (ή και 10). Τότε, σε περίπτωση που δούμε τον αλγόριθμο να σταματάει πρόωρα μπορούμε απλώς να επαναλάβουμε την εκτέλεσή του.
2. Βλέπουμε, όπως άλλωστε ήταν αναμενόμενο, το τελευταίο γονίδιο του χρωμοσώματος στην βέλτιστη λύση στο Figure 11 να ισούται περίπου με 85. Επίσης ανοίγοντας την μεταβλητή *population* από το Workspace του MATLAB είναι εμφανές ότι εν τέλη έχουμε στο 18ο γονίδιο/στήλη τιμές μικρότερες του 86 για όλα τα χρωμοσώματα. Για να βγάλουμε τον βέλτιστο χρόνο για $V = 85$ (που θεωρητικά δίνει το βέλτιστο/ελάχιστο αποτέλεσμα συνολικά για όλα τα $V \in [85, 115]$) τρέχουμε και πάλι τον κώδικα στο αρχείο `main_fmincon.m` προσέχοντας αυτήν την φορά στην συνάρτηση *initProblemValues* (μέσα στο `main_fmincon.m`) να αλλάξουμε το *vValue* σε 85. Τα αποτελέσματα παρουσιάζονται παρακάτω. Όπως βλέπουμε η δική μας λύση έχει απόκλιση από τον πραγματικό βέλτιστο χρόνο μόλις $\sim 0.8\%$.

```
Optimal Traffic Flow:
  Columns 1 through 12

   26.8701   11.8778   20.6104   25.6417   17.5503    9.3198    9.7623    2.1155   10.9388   14.7029    8.4441   17.7973

  Columns 13 through 17

    7.4233    2.2825   24.2230   19.8327   23.1469

Minimum Total Travel Time: 562.412532 min
```

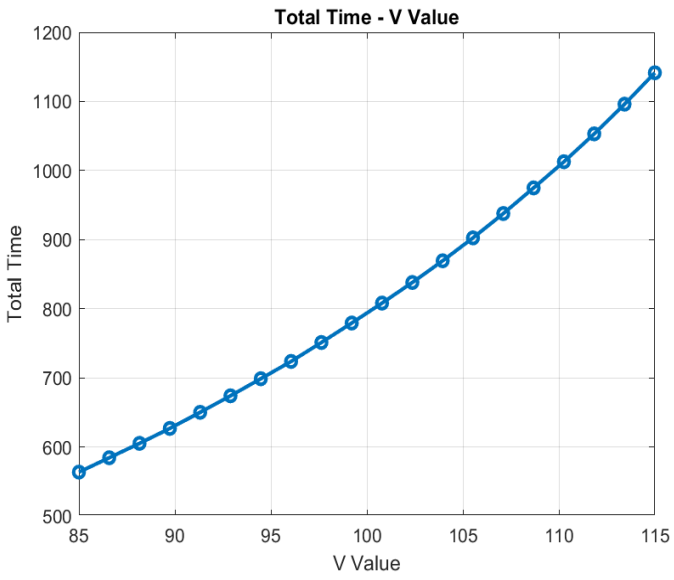
Σχήμα 12: Αποτελέσμα fmincon για $V = 85$

2ος Τρόπος: Grid Search V

Ένας ακόμη τρόπος με τον οποίο μπορούμε να βρούμε για ποιο V παίρνουμε την βέλτιστη λύση είναι να πάρουμε διάφορες τιμές του $V \in [85, 115]$ και για καθεμιά από αυτές να τρέξουμε τον γενετικό αλγόριθμο. Έτσι, θα δούμε πώς το V επηρεάζει την βέλτιστη λύση μέσω του διαγράμματος: Βέλτιστος Χρόνος - V.

Με την μέθοδο αυτή δεν χρειάζεται να εισάγουμε νέο γονίδιο στην ανάλυσή μας (μιας και το V κάθε φορά που τρέχουμε τον GA είναι σταθερό).

Για να εκτελέσουμε τον γενετικό αλγόριθμο που αναπτύξαμε στο *Θέμα 2* για 20 τιμές V ανάμεσα στο 85 και 115 (*linspace*) τρέχουμε το αρχείο: `mainGridV.m`



Από το διάγραμμα φαίνεται πως με την αύξηση του V (άρα της συνολικής κίνησης) ο συνολικός βέλτιστος χρόνος αυξάνεται με έναν μη γραμμικό τρόπο (μπορούμε να κάνουμε παλινδρόμηση με πολυώνυμο δευτέρου βαθμού).

Τέλος ο αλγόριθμος εκτυπώνει το βέλτιστο V, τον καλύτερο χρόνο και το χρωμόσωμα που οδήγησε σε αυτόν τον χρόνο για το βέλτιστο V:

```
Best *overall* solution found for V = 85
Best Total Time: 563.536787
Best Chromosome:
  Columns 1 through 12

   26.0555   12.3388   20.8381   25.7675   16.9403    9.1153    9.8556    2.4832   11.5372   14.2304    7.8337   17.5020

  Columns 13 through 17

    9.5227    4.0000   24.4936   20.9403   22.0641

Resulted Chromosome Valid !!
```

Το μειονέκτημα της παραπάνω μεθόδου - αν και μας επιτρέπει να κατανοήσουμε καλύτερα την σχέση μεταξύ βέλτιστου συνολικού χρόνου και V - είναι ότι πρόκειται για μία αργή μέθοδο καθώς επαναλαμβάνει τη διαδικασία επίλυσης για κάθε V.