UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di laurea in Informatica

UN APPROCCIO GENETICO ALL'OTTIMIZZAZIONE DELL'ORARIO SCOLASTICO

Relatore: Prof. Nicolò CESA-BIANCHI

Tesi di:
Alfonso ADDUCI
Matricola: 756947

Anno Accademico 2014-2015

A Mamma, Nonna e Nonno.

Sommario

A seguito di una breve introduzione al problema dell'ottimizzazione dell'orario scolastico ed al funzionamento degli algoritmi genetici, verrà presentata una formalizzazione del problema trattato, con il relativo modello dei dati (basato su istanze reali del sistema scolastico italiano) e la descrizione dettagliata di un algoritmo euristico per la ricerca delle soluzioni migliori. A seguire, l'analisi delle prestazioni ottenute su un caso di esempio da una sua implementazione sviluppata in linguaggio Java. Il progetto è stato realizzato nell'ambito di un apprendistato di alta formazione presso l'azienda KeyOS S.r.l., con il fine di produrre un software di natura commerciale che si occupi di elaborare orari scolastici soddisfacenti per istituti concreti.

Ringraziamenti

Desidero ringraziare il prof. Nicolò Cesa-Bianchi, relatore di questa tesi, e i proff. Bruno Apolloni e Massimiliano Goldwurm per aver scelto di prendere parte a questo progetto impegnandosi a fornirmi le ore di ore di docenza individuale necessarie.

Un sentito ringraziamento anche all'azienda KeyOS S.r.l. per aver deciso di investire su di me ed avermi dato l'opportunità di svolgere questo lavoro presso i loro uffici. In particolare ringrazio Marcello Della Monica per il continuo incoraggiamento ed i preziosi consigli.

Inoltre, non posso non ringraziare Manuel Trotta ed Alma Allevi per le numerose dritte e per tutta la pazienza dimostrata nei miei confronti, sia durante la realizzazione di questo lavoro che negli anni precedenti: non credo sia possibile esprimere a parole tutta la mia riconoscenza, vi sarò debitore a vita.

Indice

1 Introduzione

1.1 L'apprendistato di alta formazione

Il progetto presentato nelle pagine seguenti è stato svolto nell'ambito di un apprendistato di alta formazione presso l'azienda KeyOS S.r.l.

Lo scopo era quello di riprendere alcuni argomenti già trattati precedentemente dall'azienda, sfruttandone il *know-how*, aggiornandolo ed espandendolo al fine di gettare le basi per lo sviluppo di un nuovo software di natura commerciale che si occupi di trattare il problema dell'orario scolastico per le scuole italiane di ogni grado di istruzione, a partire dalle scuola primaria arrivando fino alle università.

Il contratto di *apprendistato di alta formazione* è un contratto di lavoro subordinato a tempo indeterminato finalizzato all'acquisizione di un titolo di studio, che integra la formazione pratica in azienda con la formazione universitaria. Per ulteriori approfondimenti si veda [1].

L'assunzione in apprendistato prevede la predisposizione di un *Piano Formativo Individuale* (PFI), parte integrante del contratto di lavoro, che definisce per tutta la durata del contratto il percorso formativo dell'apprendista. Nel caso di *apprendistato di alta formazione e ricerca*, nello specifico, esso consiste nel progetto formativo del corso universitario e deve essere compilato nei termini stabiliti dal CCNL di riferimento.

Il PFI che è stato attuato è il seguente:

Piano Formativo Individuale

Codice Attività formative (DM 509/99)	Insegnamento / Attività	CFU	Didattica personalizzata ed addizionale	Ore di docenza individuali
Es. B (Caratterizzanti)	Algoritmi e Strutture Dati	12	Algoritmi per l'ottimizzazione discreta. Riduzioni fra problemi di ottimizzazione. Applicazione al problema dell'orario.	15
Es. C (Affini o integrative)	Calcolo delle Probabilità e Statistica Matematica	6	Ottimizzazione stocastica ed algoritmi evolutivi.	10
Es. D (A scelta dello studente)	Ulteriori attività formative e attività sperimentali	21	Sviluppo di algoritmi evolutivi per la risoluzione del problema dell'orario; stesura dell'elaborato finale per la valutazione e validazione del progetto.	20
Es. E (Per la prova finale e per la conoscenza della lingua straniera)	PROVA FINALE	3	Realizzazione di un breve rapporto (elaborato finale) per la valutazione dell'efficacia dei contenuti didattici addizionali relativamente al progetto svolto in impresa.	5
Totale		42		25

1.1.1 Formazione specialistica aggiuntiva

Algoritmi e Strutture dati

Nel caso di *algoritmi e strutture dati*, la didattica personalizzata ha riguardato lo studio dei problemi di approssimazione con vincoli, con particolare riferimento alla classe dei problemi di scheduling di cui fa parte il problema dell'orario. In questo ambito, sono state analizzate le principali tecniche algoritmiche per la soluzione approssimata dei problemi di ottimizzazione descrivendo alcuni casi applicativi concreti. Infine, a scopo propedeutico rispetto allo studio degli algoritmi evolutivi, sono stati descritti i principali metodi di ricerca locale.

In dettaglio, il programma si è articolato come segue:

- Complessità dei problemi di ottimizzazione (4 ore)
  Richiami di complessità computazionale sui problemi di decisione. Nozione generale di PO, esempi (TSP, Graph coloring). Le classi PO, NPO, i problemi di ottimizzazione NP-hard, esempi relativi (Max clique).
- Ottimizzazione con vincoli e applicazione ai problemi di scheduling (4 ore)

- Algoritmi di approssimazione (4 ore)
  Procedure di approssimazione greedy (Knapsack, Indipendente set, TSP). Algoritmi di approssimazione per problemi di scheduling (greedy e sequenziale). Altri algoritmi sequenziali di approssimazione per problemi di partizione (Bin packing, Graph coloring).
- Ricerca locale (3 ore)
  Procedure di ricerca locale in generale. Esempi notevoli: problemi di taglio e TSP. Algoritmi euristici basati su ricerca locale.

La didattica aggiuntiva è stata integrata da un totale di 10 ore di tutoraggio accademico in presenza, il cui scopo è stato supportare l'apprendimento e armonizzare la formazione accademica con quella aziendale.

Calcolo delle Probabilità e Statistica Matematica

Per quanto riguarda *calcolo delle probabilità e statistica matematica*, la didattica personalizzata ha introdotto alcuni concetti essenziali per l'attuazione del piano formativo individuale, che di norma non trovano posto nel programma regolare di studio. In particolare, sono stati forniti alcuni cenni riguardanti i processi stocastici markoviani per poi sviluppare il tema generale dell'ottimizzazione stocastica mediante algoritmi evolutivi. Il tema è stato poi calato in un contesto concreto attraverso lo studio di un problema di scheduling. Infine, sono state presentate due fra le principali tecniche di ottimizzazione stocastica: l'annealing simulato e - nell'ambito della programmazione evolutiva - gli algoritmi genetici.

- Catene di Markov (2 ore)
- Ottimizzazione stocastica e algoritmi evolutivi (1 ora)
- Caso di studio: un problema di scheduling (1 ora)
- Annealing simulato (3 ore)
- Algoritmi genetici (3 ore)

La didattica aggiuntiva è stata integrata da un totale di 15 ore di tutoraggio accademico in presenza, articolate come segue:

- Introduzione all'utilizzo di Mathematica (5 ore)
- Implementazione di un algoritmo di Simulated Annealing (5 ore)
- Implementazione di un algoritmo genetico (5 ore)

1.2 Il problema dell'orario scolastico

1.2.1 Descrizione

Il problema dell'orario scolastico fa parte dei problemi di *timetabling*, e come tale può essere definito come la pianificazione di un certo numero di attività (lezioni) a cui partecipano uno specifico gruppo di persone (studenti, docenti) per un definito periodo di tempo, che richiedono certe risorse (aule, laboratori, palestre, ecc.) in conformità con la disponibilità delle stesse e soddisfacendo certi altri requisiti [2].

Esso risulta essere una sfida particolarmente stimolante dal momento che per la sua risoluzione non si conoscono algoritmi risolutivi deterministici che impiegano un tempo polinomiale. Ciò rende necessario l'utilizzo di tecniche di ricerca avanzate che fanno uso di diverse euristiche al fine di escludere dallo spazio di ricerca quelle regioni in cui non è garantito trovare delle buone soluzioni.

Spesso i problemi di *timetabling* aventi consistenza reale sembrano non essere particolarmente difficili da risolvere se si prendono in considerazione solo i vincoli di base; questo è quanto viene spesso fatto nella risoluzione manuale dei problemi dell'orario di diverse istituzioni. Ma le soluzioni prodotte spesso risultano "cattive" perché non tengono conto dei vincoli d'ampiezza degli eventi o altri vincoli la cui gestione può risultare fastidiosa [3].

Benché infatti la compilazione di un orario attuabile possa essere facilmente eseguita attraverso l'utilizzo di un algoritmo greedy, la ricerca di un orario ottimo, che tenga cioè in considerazione tutti gli altri vincoli classificati come vincoli flessibili pure essendo fortemente desiderabili, rendono il problema un problema molto complesso di ottimizzazione multi-obiettivo, la cui risoluzione può richiedere considerevoli sforzi.

1.2.2 Complessità computazionale

Coloro che si occupano della pianificazione dell'orario all'interno di un istituto sono interessati ad un algoritmo che si occupi di trovare una soluzione ottima o quanto più possibile prossima alla soluzione ottima per un'istanza del problema. Anche solo dimostrare l'esistenza di tale soluzione, infatti, risulta essere un compito non banale.

La teoria della complessità computazionale fornisce prova del fatto che trovare una soluzione a questo genere di problemi potrebbe richiedere dei tempi di esecuzione inaccettabilmente lunghi.

Formalmente, infatti, il problema dell'ottimizzazione dell'orario viene classificato come un problema della classe NP-hard (o NP-difficile). La dimostrazione può essere effettuata prendendo in considerazione la sua variante decisionale (che consiste nel determinare l'esistenza o meno di una soluzione con una certa *misura*, data una certa istanza) e mostrando che questa costituisce un problema NP-completo, ossia che esiste almeno un problema NP-completo polinomialmente *Karp-riducibile* ad esso (tipicamente viene utilizzato a tale scopo il problema Satisfiability, *SAT*). In [2] è possibile osservare la dimostrazione completa per un problema analogo a quello affrontato in questa tesi.

È per questo motivo che si presenta la necessità di utilizzare un algoritmo euristico, il cui obiettivo è produrre una soluzione ragionevolmente buona in una finestra temporale limitata.

1.3 Gli algoritmi genetici

Gli algoritmi genetici sono una classe di algoritmi di ottimizzazione stocastica ispirati dalla biologia, ed in particolare da quei processi biologici che consentono alle popolazioni di organismi di adattarsi all'ambiente che li circonda: l'ereditarietà genetica e la selezione naturale.

Essi vengono solitamente adoperati per la ricerca di soluzioni ottimali a problemi complessi, in cui la funzione obiettivo è discontinua e non lineare, per i quali è inefficace o dispendioso l'utilizzo degli algoritmi lineari classici. Gli algoritmi genetici non assicurano l'individuazione di una soluzione ottimale ma contribuiscono ad individuare un insieme di soluzioni migliori rispetto alle soluzioni di partenza. A partire da un medesimo problema e da un medesimo insieme di soluzioni possibili di partenza, ad ogni esecuzione questi algoritmi possono evolvere verso soluzioni finali differenti [4].

1.3.1 Il cuore dell'algoritmo

Un algoritmo genetico mantiene una popolazione di soluzioni candidate per il problema trattato, e le fa evolvere applicando iterativamente un insieme di operatori stocastici quali *mutazione*, *ricombinazione* e *selezione*.

- La *mutazione* genera una perturbazione casuale all'interno di una soluzione candidata;
- la *ricombinazione* scompone due soluzioni distinte, ed a partire dalle loro parti compone in maniera casuale una nuova soluzione;
- la *selezione* replica le soluzioni migliori di una popolazione con un tasso proporzionale alla loro qualità.

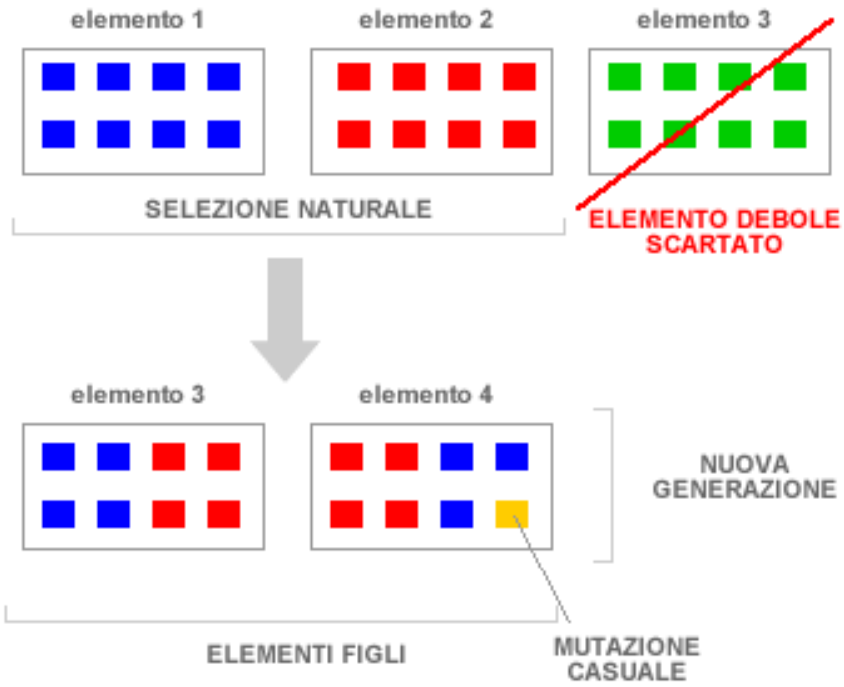

FIGURA 1.1 MUTAZIONE, RICOMBINAZIONE E SELEZIONE IN UN ALGORITMO GENETICO

1.3.2 Lo schema generale

L'evoluzione delle soluzioni avviene tipicamente secondo il seguente schema [5]:

1. Viene formata la popolazione iniziale con un campione casuale di soluzioni dette *cromosomi*.
2. Ad ogni soluzione viene associato un certo grado di valutazione attraverso una funzione appositamente progettata detta funzione di *fitness*.
3. Vengono ripetuti i seguenti passi tante volte quanti sono gli individui da cui è formata una popolazione

a. Si seleziona una coppia di cromosomi genitori dalla popolazione attuale, con una probabilità proporzionale al valore di fitness. La selezione avviene *con reinserimento*, ovvero alcuni cromosomi possono essere selezionati per diventare genitori più di una volta.
b. Viene generata una nuova soluzione a partire dalla coppia individuata. Il procedimento avviene mescolando le caratteristiche (*geni*) di entrambi gli individui, attraverso la funzione di *crossover*.
c. In alcune nuove soluzioni viene generata (con una certa probabilità) un'alterazione casuale detta *mutazione*.

4. Si rimpiazza la popolazione corrente con quella appena generata.
5. Il processo viene reiterato a partire dal punto 2.

Il processo risultante - se eseguito per un tempo sufficiente - tende a trovare soluzioni globalmente ottime al problema in maniera molto simile al modo in cui in natura le popolazioni di organismi tendono ad adattarsi all'ambiente che li circonda.

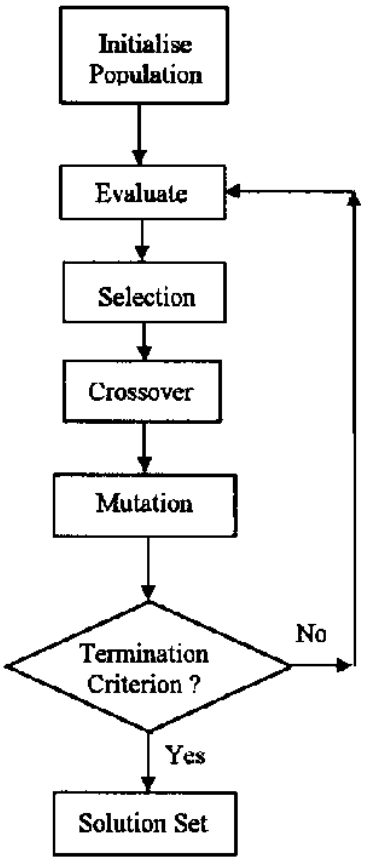

FIGURA 1.2 SCHEMA GENERALE DEL FUNZIONAMENTO DI UN ALGORITMO GENETICO

2 Modello dei dati

Al fine di agevolare l'interoperabilità ed accentuare l'indipendenza tra l'interfaccia utente di gestione dei dati ed il software che si occupa di eseguire l'algoritmo per la ricerca delle soluzioni, i dati sono stati trattati con un *DBMS relazionale*, nello specifico *MySQL*.

2.1 Modello concettuale

Il modello presentato è stato ricavato a partire dall'osservazione di una serie di istituti scolastici italiani di ogni ciclo di istruzione. È stato utilizzato un approccio di tipo *bottom-up*, partendo cioè dalle entità individuate e descritte nel paragrafo seguente e procedendo poi all'analisi del modo in cui esse si relazionano ed all'estrapolazione dei loro attributi.

2.1.1 Entità

D è l'insieme dei **docenti**;

Cl è l'insieme delle **classi** a cui saranno impartite le lezioni;

M è l'insieme delle **materie**;

A è l'insieme delle **aule**;

S è l'insieme delle **sedi** (edifici) dell'istituto;

T è l'insieme dei **tipi di aule** (es. ordinaria, palestra, laboratorio informatico, ecc.);

P è l'insieme dei **periodi**, ovvero delle unità temporali in cui una lezione può essere svolta.

2.1.2 Relazioni

Un **corso** è definito come una terna ordinata *Co: (d, c, m)*, dove:

- $d \in D$ è il docente che tiene il corso;
- $c \in Cl$ è la classe a che segue il corso;
- $m \in M$ è la materia oggetto del corso.

Una **lezione** è definita come una terna ordinata *L: (a ,c, p)*, dove:

- $a \in A$ è l'aula in cui si svolge la lezione;
- $c \in Co$ è il corso a cui la lezione appartiene;
- $p \in P$ è l'unità di tempo (periodo) in cui la lezione viene svolta.

Un'**aula è associata ai suoi tipi** attraverso la funzione

$$tipiAula: A \rightarrow T$$

che associa ad ogni aula le caratteristiche $t \in T$ che essa possiede.

Una **materia è associata ai suoi tipi richiesti** attraverso la funzione

$$tipiMateria: M \rightarrow T$$

che associa ad ogni materia l'insieme delle caratteristiche $t \in T$ da essa richieste richieste per l'aula in cui deve essere insegnata.

La **disponibilità** dei docenti è definita dalla funzione

$$disponibilità: D \times P \rightarrow \{0,1,2,3\}$$

che applicata ad una coppia ordinata $(d, p) \in D \times P$, indica con un numero naturale il grado di disponibilità del docente d nel periodo p. I gradi di disponibilità rispetto ad un certo periodo sono mappati come segue:

valori di disponibilità

Valore	Descrizione
0	gradito
1	disponibile
2	sgradito
3	indisponibile

Ogni **aula è associata alla sua sede** di appartenenza attraverso la funzione

$$sede: A \to S$$

2.1.3 Attributi

Ad ogni corso $c \in Co$ è associato attraverso la funzione

$$periodiSettimanali: Co \to \mathbb{N}$$

un certo **numero di periodi** $p \in P$ che devono essergli assegnati settimanalmente.

Ad ogni classe è associato un **numero di alunni** attraverso la funzione

$$alunni: Cl \to \mathbb{N}.$$

Ad ogni aula è associata una **capienza** attraverso la funzione

$$capienza: A \to \mathbb{N}$$

che indica il numero di alunni che essa può contenere.

Ad ogni periodo si associa una determinata **ora del giorno** a cui esso corrisponde attraverso la funzione

$$ora: P \to \mathbb{N}$$

che restituisce 1 per la 1ª ora, 2 per la 2ª ora, ecc.

Ogni periodo è associato ad un determinato **giorno della settimana** attraverso la funzione

$$giorno: P \to \mathbb{N}$$

che restituisce 1 per il lunedì, 2 per il martedì, ecc.

2.2 Modello logico

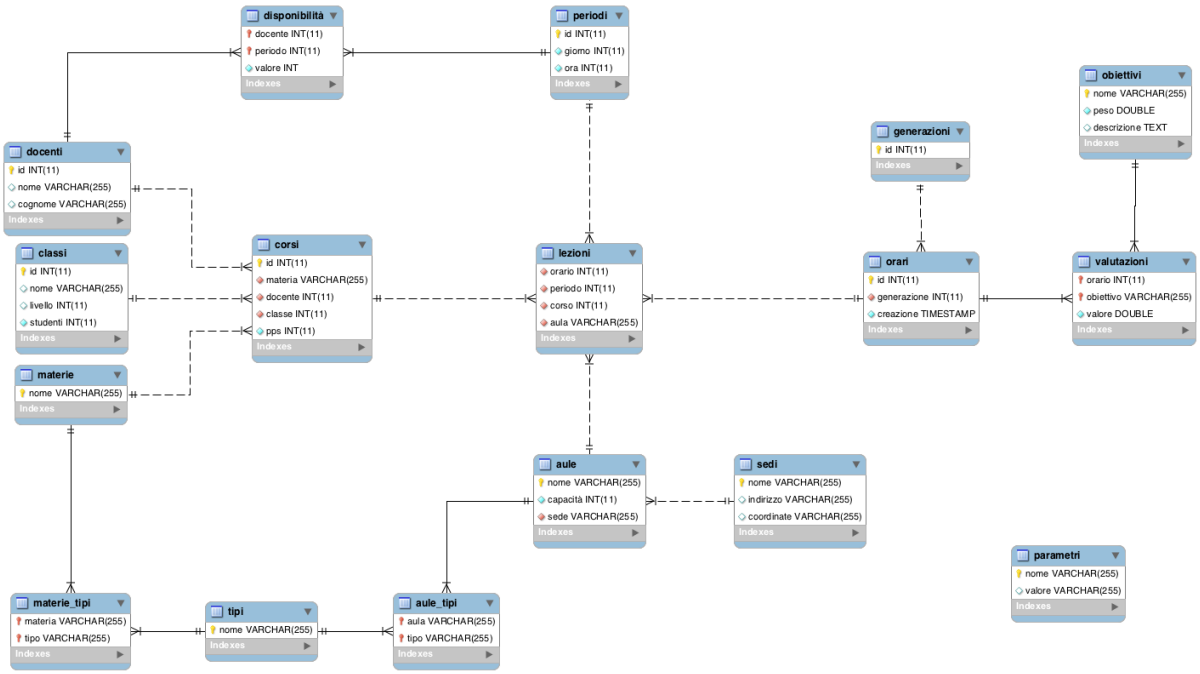

FIGURA 2.1 DIAGRAMMA EER DELLA BASE DI DATI UTILIZZATA

Per la realizzazione dello schema relazionale sono state utilizzate le comuni regole di derivazione, nello specifico:

- Ogni entità genera una tabella;
- Le relazioni da uno a molti vengono gestite tramite una chiave esterna nella tabella correlata;
- Le relazioni da molti a molti generano una tabella;
- Ogni attributo di un'entità corrisponde ad un campo nella struttura del record della relativa tabella.

Si hanno pertanto le seguenti tabelle:

tabella docenti

Nome	Tipo	Altro
id	INT	PK
nome	CHAR	
cognome	CHAR	

tabella classi

Nome	Tipo	Altro
id	INT	PK
nome	CHAR	
livello	CHAR	
studenti	INT	

tabella materie

Nome	Tipo	Altro
nome	CHAR	PK

tabella corsi

Nome	Tipo	Altro
id	INT	PK
docente	INT	FK
classe	INT	FK
materia	INT	FK

tabella periodi

Nome	Tipo	Altro
id	INT	PK
giorno	INT	
ora	INT	

tabella sedi

Nome	Tipo	Altro
nome	CHAR	PK
indirizzo	CHAR	
coordinate	CHAR	

tabella aule

Nome	Tipo	Altro
nome	CHAR	PK
capienza	INT	
sede	CHAR	FK

tabella tipi

Nome	Tipo	Altro
nome	CHAR	PK

tabella disponibilità

Nome	Tipo	Altro
docente	INT	PK, FK
periodo	INT	FK, FK
valore	INT	

tabella materie_tipi

Nome	Tipo	Altro
materia	CHAR	PK, FK
tipo	CHAR	PK, FK

tabella aule_tipi

Nome	Tipo	Altro
aula	CHAR	PK, FK
tipo	CHAR	PK, FK

tabella lezioni

Nome	Tipo	Altro
orario	INT	FK
periodo	INT	PK, FK
corso	INT	PK, FK
aula	CHAR	PK, FK

Inoltre, per poter gestire i dati relativi al funzionamento dell'algoritmo evolutivo e permetterne in qualsiasi momento il tracciamento, l'interruzione ed eventualmente la ripresa, sono state create le seguenti tabelle:

Ogni lezione appartiene ad un **orario**, che come verrà successivamente spiegato, costituisce la soluzione del problema.

tabella orari

Nome	Tipo	Altro
id	INT	PK
generazione	INT	FK
creazione	DATETIME	

Ogni orario appartiene ad una **generazione**.

tabella generazioni

Nome	Tipo	Altro
id	INT	PK

Un **obiettivo** indica un vincolo soft, ed ogni obiettivo può avere un certo peso in base alle preferenze dell'istituto.

tabella obiettivi

Nome	Tipo	Altro
nome	CHAR	PK
peso	DOUBLE	
descrizione	CHAR	

Una **valutazione** rappresenta il punteggio ottenuto da un orario rispetto ad un determinato obiettivo.

tabella valutazioni

Campo	Tipo	Altro
orario	INT	PK, FK
obiettivo	CHAR	PK, FK
valore	DOUBLE	

È necessario infine tenere conto di un insieme di **parametri** nome-valore che regolano il funzionamento dell'algoritmo e del programma.

tabella parametri

Campo	Tipo	Altro
nome	CHAR	PK
valore	CHAR	

3 Formalizzazione del problema

Come ogni problema di ottimizzazione combinatoria, il problema dell'orario scolastico può essere definito come una quadrupla (I, f, m, g), dove:

- I è un insieme di **istanze**;
- data un'istanza $x \in I$, $f(x)$ è l'insieme delle **soluzioni fattibili**;
- data un'istanza x ed una soluzione fattibile y di x, $m(x,y)$ denota la **misura** di y;
- $g \in \{min,max\}$ è la funzione di **scopo** (goal).

3.1 Istanze del problema

Le istanze sono definite da una terna ordinata $I: (E, R, A)$, dove:

- $E=\{D, Cl, M, A, S, T, P\}$ è una famiglia di **entità**;
- $R=\{Co, tipiAula, tipiMateria, disponibilità, sede\}$ è una famiglia di **relazioni**;
- $A=\{periodiSettimanali, alunni, capienza, ora, giorno\}$ è una famiglia di **attributi**.

3.2 Soluzioni

Una soluzione è data da un **orario**, ovvero famiglia O di lezioni $l \in L$ che rispettano i **vincoli hard** $V_h=\{1, 2, 3, 4, 5, 6\}$.

1. Le lezioni rispettano il numero di ore previste per ogni corso:
   $\forall\ c \in Co$
   $|\{(a, c, p) \mid (a, c, p) \in O\}| = periodiSettimanali(c)$

2. Ogni aula in un periodo prevede al più una lezione:
   $\forall\ a \in A, \forall\ p \in P$
   $(a, c', p) \in O \wedge (a, c'', p) \in O \Rightarrow c'=c''$

3. Ogni classe in un periodo assiste al più ad una lezione:
   $\forall\ c \in Cl, \forall\ p \in P$
   $(a', (i', c, m'), p) \in O \wedge (a'', (i'', c, m''), p) \in O \Rightarrow a'=a'' \wedge d'=d'' \wedge m'=m''$

4. Ogni docente in un periodo tiene al più una lezione:
   $\forall\ d \in D, \forall\ p \in P$
   $(a', (d, c', m'), p) \in O \wedge (a'', (d, c'', m''), p) \in O \Rightarrow a'=a'' \wedge c'=c'' \wedge m'=m''$

5. Ogni lezione si svolge in aule adeguate:
   $\forall\ (a, (d, c, m), p) \in O$
   $tipiMateria(m) \subseteq tipiAula(a)$

6. Le lezioni si svolgono in aule abbastanza capienti:
   $\forall\ (a, (d, c, m), p) \in O$
   $alunni(c) \leq capienza(a)$

Il numero di lezioni da programmare è dato da

$$\sum_{c \in Co} periodi\_settimanali(c)$$

3.3 Funzione di misura

La funzione di misura si occupa di associare ad ogni soluzione un valore numerico che ne esprima la bontà. In questo caso, per poter valutare la bontà di un orario scolastico è necessario definire una famiglia di **vincoli soft** $V_s=\{1,2,3\}$, ovvero di vincoli che pur essendo fortemente desiderabili possono essere violati qualora necessario.

1. Due lezioni di uno stesso corso, se tenute lo stesso giorno dovrebbero essere consecutive:
   $\forall$ *(a', c', p'), (a'', c'', p'')* $\in O$
   giorno(p')=giorno(p'') $\wedge$ *c'=c''* $\Rightarrow$
   $\forall$ $p \in P$ *t.c. giorno(p')=giorno(p)=giorno(p'')* $\wedge$ *ora(p')<ora(p)<ora(p'')*
   se (a, c, p) $\in O$ *allora a=a'=a''* $\wedge$ *c=c'=c''*

2. Le lezioni di un corso devono essere distribuite uniformemente nella settimana: sia *lezioniGiornaliere: Co×giorno(P)* $\rightarrow \mathbb{N}$ la funzione che associa ad una coppia ordinata (c, g) con c $\in$ *Co* e $g \in$ *giorno(P)* al numero di lezioni $l \in L$ del corso c nel giorno g, allora
   $\forall$ $c \in$ *Co*, $\forall$ $g \in$ *giorno(P)*
   lezioniGiornaliere(c, g) <= $\lceil$*periodiSettimanali(c) / |giorno(P)|*$\rceil$

3. Ogni docente dovrebbe insegnare solo nelle ore che gli sono gradite:
   $\forall$ *(a, (d, c, m), p)* $\in O$
   disponibilità(d, p)=0

Come si può notare, l'ottimizzazione di un orario scolastico è un problema di natura multi-obiettivo. Le funzioni di misura da ottimizzare sono infatti tante quanti sono i vincoli soft e sono costituite dal "*grado di rottura*" di ogni vincolo, espresso dalla funzione:

$$gradoRottura: V_s \times O \rightarrow [0,1]$$

che associa ad una coppia ordinata *(v,o)* con $v \in V_s$ e $o \in O$ un punteggio per l'orario o relativamente al vincolo v.

Un modo per affrontare questo genere di situazioni è quello di combinare le varie funzioni di misura al fine di produrre una *funzione di misura scalare aggregata* [6].

Nei casi 1 e 2, ciò equivale al nel semplice conteggio delle violazioni presenti nell'orario o, in relazione al numero massimo di violazioni possibili:

1. $|\{(a', c', p') \in O \mid \exists\ (a'', c'', p'') \in O$ *t.c.* $giorno(p')=giorno(p'') \wedge c'=c'' \wedge$
   $\exists\ (a, c, p) \in O$ *t.c.* $giorno(p')=giorno(p)=giorno(p'') \wedge$
   $ora(p')<ora(p)<ora(p'') \wedge$
   $a \neq a'=a'' \vee c \neq c'=c''\}|$

2. sia $lezioniGiornaliere: Co \times giorno(P) \rightarrow \mathbb{N}$ la funzione che associa ad una coppia ordinata (c, g) con c $\in Co$ e $g \in giorno(P)$ al numero di lezioni $l \in L$ del corso c nel giorno g, allora
   SUM for (c,g) $\in Co \times giorno(P)$ *of*
   $min\{0, \lceil periodiSettimanali(c) / |giorno(P)| \rceil - lezioniGiornaliere(c, g)\}$

Mentre, nel caso 3, viene utilizzata la sommatoria dei gradi di disponibilità di ogni professore per ogni lezione della soluzione

3. $\sum_{(a,(d,c,m),p) \in O} disponibilità(d,p)$

Per normalizzare i valori ottenuti, però, ovvero fare in modo che essi cadano nell'intervallo *[0,1]* (in cui 0 indica il soddisfacimento pieno della condizione), sarà necessario metterli in rapporto con il grado di rottura massimo, che si valuterà considerando ogni lezione come una violazione nei casi 1 e 2, e valutando ogni lezione con il grado di disponibilità peggiore nel caso 3.

La scelta di limitare ad un intervallo chiuso la valutazione dei gradi di rottura è stata fatta al fine di uniformare i valori calcolati per i diversi vincoli. Questa condizione permetterà di fare in modo che sia l'utente ad esprimere una preferenza per ognuno di essi, assegnando ad ogni vincolo una costante (che chiameremo moltiplicatore o peso) in base alla desiderabilità dell'obiettivo corrispondente.

Definita dunque la funzione

$$moltiplicatore: V_s \rightarrow \mathbb{R}^+$$

che associa ad ogni vincolo $v \in V_s$ il moltiplicatore impostato dall'utente. La formula per calcolare la bontà di una soluzione $o \in O$, ovvero la funzione di misura, risulta infine essere una combinazione lineare nella forma:

$$m(o) = \sum_{v \in V_s} gradoRottura(v,o) \cdot moltiplicatore(v)$$

3.4 Scopo

Chiaramente, una *soluzione ottima* è data da un orario tale per cui tutti i vincoli soft hanno grado di rottura pari a 0. Di conseguenza, l'insieme delle soluzioni ottime è costituito dagli orari per cui la funzione di fitness restituisce 0, ovvero

$$f^*(x) = \{o \in O \mid m(x) = 0\}.$$

Più in generale, comunque, minore è il grado di rottura per ogni vincolo $v \in V_s$, minore è il valore restituito dalla funzione di misura e migliore è la soluzione.

Ciò significa che lo scopo è quello di *minimizzare* la funzione di misura, ovvero:

$$g = min.$$

4 Un algoritmo genetico per la ricerca della soluzione ottima

Lo schema generale di funzionamento dell'algoritmo realizzato, i cui dettagli implementativi verranno discussi in questo capitolo, è il seguente:

```
Procedura cercaOrario(popSize,tournamentSize,pCross,pMutate,mutationSize)
```

```
begin
   popolazione := inizializzaPopolazione(popSize)
   soluzione := Λ

   while !termina() do
      for orario ∈ popolazione do
         if
            IS_EMPTY(soluzione) ∨
            calcolaFitness(orario) < calcolaFitness(soluzione)
         then
            orario := soluzione
         end if
      end for

      vacchiaPopolazione := popolazione
      popolazione := Λ

      for i := 1 to popSize do
         genitori := seleziona(vacchiaPopolazione, tournamentSize, pCross)
         figlio := incrocia(ELEMENTO(genitori, 1), ELEMENTO(genitori, 2))

         if random() < pMutate do
            figlio := mutazioneIntelligente(figlio, mutationSize)
         end if
      end for

   end while

   return orario
end
```

I parametri *popSize, pMutate e mutationSize* definiscono rispettivamente il numero di individui di cui è composta una popolazione, la probabilità di mutazione per ogni nuovo elemento generato e la dimensione di tale mutazione.

I parametri *tournamentSize* e *pCross*, invece, agiscono sulla logica di selezione e vanno di fatto a definire la probabilità con cui un individuo verrà selezionato per l'accoppiamento.

Inoltre:

- *termina()*: è una funzione che restituisce *true* quando vengono soddisfatte determinate condizioni di terminazione, che sono solitamente determinate dal budget temporale a disposizione o al valore minimo di bontà della soluzione che si desidera raggiungere. Altre possibilità sono l'arresto manuale da parte dell'utente o la valutazione dei miglioramenti ottenuti in un certo lasso di tempo, ad esempio si potrebbe decidere di arrestare l'esecuzione se non sono state trovate soluzioni migliori nell'arco degli ultimi 10 minuti di esecuzione, o nelle ultime 100 iterazioni dell'algoritmo.
- *random()*: è una funzione che restituisce un numero casuale nell'intervallo [0,1].

4.1 Codifica

Tradizionalmente, in un algoritmo genetico le soluzioni vengono rappresentate come una stringa di 0 e 1 in cui ogni bit esprime una caratteristica elementare della soluzione, in quella che è detta *codifica vettoriale binaria* [7].

In questo caso invece è stata scelta per il genotipo un tipo di rappresentazione che nella letteratura è denominata *codifica vettoriale diretta*: in questo tipo di codifica ogni campo contiene direttamente i valori relativi al problema, tecnica che ben si presta a problemi altamente specifici quali possono essere quelli derivati da istanze del sistema scolastico italiano, e presenta ulteriori vantaggi non trascurabili dal punto di vista della codifica.

Questa opzione infatti è risultata la più appropriata non solo a causa della complessità intrinseca delle soluzioni, ma anche perché la progettazione di una funzione di fitness significativa, così come quella di una funzione riparazione, avrebbe in ogni caso richiesto la decodifica dei dati.

Con questo tipo di rappresentazione ogni individuo della popolazione risulta definito da una lista contenente un numero di lezioni pari al numero di tutte le lezioni della scuola che devono essere pianificate.

L'algoritmo è stato codificato in linguaggio *Java*. L'Object-Relational *Mapping (ORM)* è stato realizzato utilizzando il framework *Hibernate*. Le classi utilizzate hanno una corrispondenza diretta con le tabelle del modello logico descritto nel capitolo 2.2, essendo state ricavate automaticamente dallo stesso.

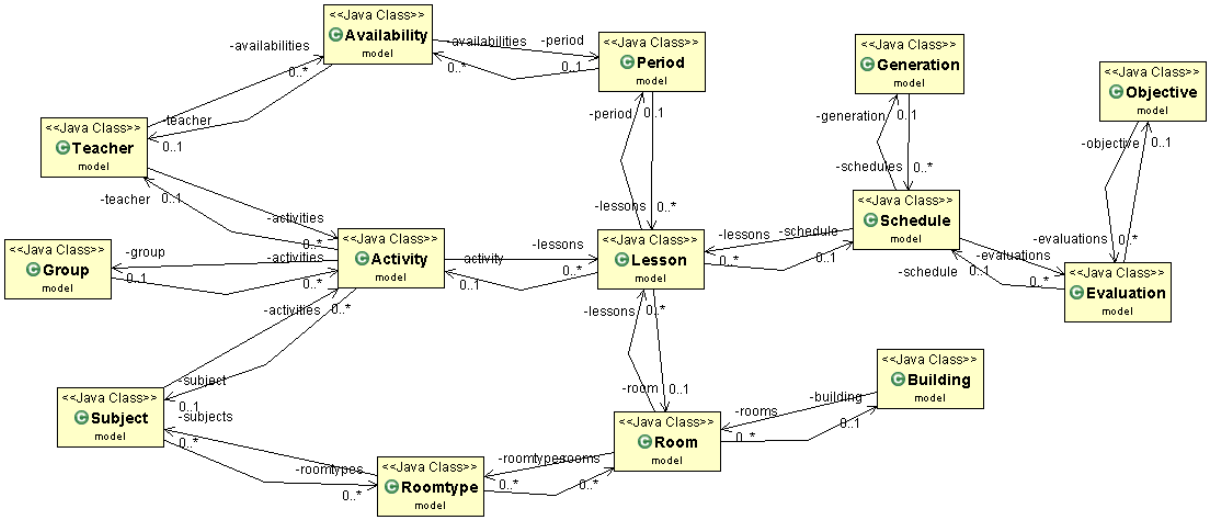

FIGURA 4.1 CLASS DIAGRAM DELL'IMPLEMENTAZIONE REALIZZATA IN JAVA

Una soluzione, quindi, è rappresentata dalla classe *Schedule*, che contiene una lista di *Lesson*.

4.2 Generazione della popolazione iniziale

L'utente, nella fase iniziale, si occupa di definire l'intera istanza del problema. Come esposto in dettaglio nel capitolo 3.1, ciò consiste nel determinare l'insieme delle risorse scolastiche, il modo in cui esse interagiscono e una serie di serie di proprietà quali i giorni di insegnamento, il numero lezioni programmabili per ogni giorno e la disponibilità dei docenti.

Il software si occupa di dedurre una lista di lezioni da allocare: vengono associati docenti, aule, classi e materie in maniera conforme ai vincoli, e ad esse dovrà poi soltanto essere assegnato un periodo di svolgimento.

La generazione delle lezioni da allocare viene eseguita attraverso un algoritmo greedy che per ogni corso, e per ogni periodo di docenza previsto dallo stesso, genera una lezione e vi associa un'aula idonea.

```
Procedura getLezioniDaAllocare()
```

```
begin
   lezioniDaAllocare:=Λ

   for c ∈ Co do

      for i := 1 to periodiSettimanali(c) do
         a := ottieniAula(c,lezioniDaAllocare)
         l := (a, c, -)

         INSERISCI_IN_TESTA(lezioniDaAllocare, l)
      end for

   end for

   return lezioniDaAllocare
end
```

La procedura *cercaAula* restituisce, in base ad un certo corso ed alla lista delle lezioni già allocate, l'aula più opportuna in cui svolgere la lezione. Si occupa di trovare la lista delle aule che soddisfano tutte le caratteristiche richieste per l'insegnamento della materia e che abbiano una capienza adeguata. Inoltre, nel caso vi siano altre lezioni della stessa classe già allocate, cerca di utilizzare la stessa aula (ammesso che abbia ancora periodi disponibili), questo al fine di ridurre gli spostamenti.

Nel dettaglio:

```
Procedura ottieniAula(corso, lezioniDaAllocare)
```

```
begin
   for x ∈ {(a,c,p) ∈ lezioniDaAllocare | c=corso} do
      if  |{(a,c,p) ∈ lezioniDaAllocare | a=x}| < |P| then
         return x
      end if

      for x ∈ Cl do
         if tipiMateria(c) ⊆ tipiAula(x) then
            return x
         end if
      end for
end
```

P rappresenta l'insieme dei periodi d'insegnamento.

Inoltre, in accordo con quanto detto nel capitolo 2.1.2:

- *tipiMateria(corso)*: prende in input un corso e restituisce l'insieme delle caratteristiche che la materia trattata richiede per poter essere insegnata in un'aula.

- *tipiAula(aula)*: prende in input un'aula e restituisce la lista delle caratteristiche che essa possiede.

A questo punto per ottenere una soluzione accettabile non resta che assegnare dei periodi in ordine casuale, avendo cura che rispettino i *vincoli hard*.

Questo viene fatto attraverso la funzione assegnaPeriodi:

```
Procedure assegnaPeriodi(lezioniAllocate, lezioniDaAllocare)
```

```
begin
   while !IS_EMPTY(lezioniDaAllocare) do
      lezione := TESTA(lezioniDaAllocare)
      lezioniDaAllocare := TOGLI(lezioniDaAllocare,lezione)

      if assegnaPeriodo(lezione) then
         lezioniAllocate := INSERISCI_IN_CODA(lezioniAllocate, lezione)
      else
         lezioniDaAllocare := INSERISCI_IN_CODA(lezioniDaAllocare, lezione)
         lezioneSimile := getLezioneSimile(lezione, lezioniAllocate)
         lezioniAllocate := TOGLI(lezioniAllocate, lezioneSimile)
         togliPeriodo(lezioneSimile);
         INSERISCI_IN_CODA(lezioniDaAllocare,lezioneSimile)
      end if
   end while

   return lezioniAllocate;
end
```

In cui:

- *assegnaPeriodo(lezione)*: è una funzione che prende in input il riferimento ad una lezione (a,c,-) e vi assegna un periodo p in maniera da che non venga rotto nessun vincolo hard. Se l'operazione va a buon fine restituisce *true*, altrimenti, nel caso non dovessero esserci periodi idonei, restituisce *false*.
- *togliPeriodo(lezione)*: è una funzione che prende in input il riferimento ad una lezione (a,c,p) e vi rimuove il riferimento al periodo di svolgimento p.
- *getLezioneSimile(lezione, lezioniAllocate)*: è una funzione che prende in input una particolare lezione e la lista delle lezioni allocate, e restituisce una lezione appartenente alla lista delle lezioni allocate che abbia lo stesso docente, la stessa aula o lo stesso periodo di svolgimento della lezione fornita in input.

Infine, dunque, ripetendo questa procedura per *popSize* volte (ovvero tante volte quanti sono gli individui di una popolazione) si potrà completare l'inizializzazione della popolazione iniziale. In dettaglio:

```
Procedura inizializzaPopolazione(popSize):
```

```
begin
   popolazione := Λ
   lezioniDaAllocare := getLezioniDaAllocare(I)

   for i := 1 to popSize do
      lezioniDaAllocare := disordina(lezioniDaAllocare)
      individuo := assegnaPeriodi(Λ ,lezioniDaAllocare)
      INSERISCI_IN_TESTA(popolazione, individuo)
   end for

   return popolazione
end
```

4.3 Funzione di fitness

Nella metafora degli algoritmi genetici, la funzione di fitness rappresenta l'adeguatezza di un individuo alla sopravvivenza ed alla riproduzione. Un individuo con una fitness maggiore dovrebbe dunque rappresentare una soluzione migliore.

Come si può notare, questo concetto è molto simile a quanto è stato detto a proposito della funzione di misura (o funzione obiettivo), se non fosse che trattandosi di un problema di minimizzazione, una soluzione migliore è rappresentata da un valore minore.

Questa imprecisione linguistica si potrebbe risolvere definendo la funzione di fitness come la funzione di misura cambiata di segno: in questo modo si manterrebbe il vantaggio di conoscere a priori il valore di una soluzione ottima a prescindere dall'istanza (0), e si trasformerebbe il tutto in un problema di massimizzazione, lavorando su valori di fitness negativi.

Tuttavia, dato che l'overhead aggiunto, seppur minimo, non sarebbe giustificato da alcun reale vantaggio, questa ipotesi è stata scartata: la funzione di fitness dunque coinciderà esattamente con la funzione di misura già definita.

Essa non farà altro che eseguire la somma dei gradi di rottura per ogni vincolo soft, moltiplicandoli per le costanti definite dall'utente. La descrizione in pseudo-codice è la seguente:

```
Procedura calcolaFitness(schedule, vincoliSoft)
```

```
begin
   fitness := 0
   for v ∈ vincoliSoft do
      fitness := fitness + (gradoRottura(v,schedule) * moltiplicatore(v))
   end for

   return fitness
end
```

In cui:

- *gradoRottura(vincolo, schedule)*: esegue la valutazione del grado di rottura secondo le regole descritte nel paragrafo 3.3.

4.4 Logica di selezione

A causa di complessi fenomeni di interazione non lineare (epistaticità), non è dato per scontato né che da due soluzioni promettenti ne nasca una terza più promettente né che da due soluzioni con valori di fitness basso ne venga generata una terza con valore di fitness più basso. Per ovviare a questo problema, durante la scelta delle soluzioni candidate all'evoluzione, è necessario fare in modo che anche a soluzioni che non abbiano il massimo valore di fitness possa capitare di essere selezionate per l'accoppiamento.

È stato scelto di utilizzare un meccanismo che in letteratura è conosciuto come "*selezione a torneo*", ovvero:

- Si scelgono in maniera casuale *tournamentSize* individui appartenenti alla popolazione.
- Si sceglie l'individuo migliore e si imposta la sua probabilità di scelta a *pCross.*
- Si scegliere il secondo individuo migliore e si imposta la sua probabilità di scelta a $p \cdot (1 - pCross)$.
- Si sceglie il terzo individuo migliore e si imposta la sua probabilità di scelta a $p \cdot (1 - pCross)^2$.
- [...]
- Si selezionano due individui per la riproduzione, rispettando le probabilità assegnate.

La procedura risultante è:

```
Procedura seleziona(popolazione, torunamentSize, pCross)
```

```
begin
   partecipanti := Λ
   selezione := Λ
   i := 0

   popolazione = disordina(popolazione)

   for i := 1 to torunamentSize do
      INSERISCI_IN_TESTA(partencipanti, ELEMENTO(popolazione, i))
   end for

   ordinaPerFitness(partecipanti)

   while LUNGHEZZA(selezione) < 2 do
      i= (i+1) % LUNGHEZZA(partecipanti)
      if random() < pCross then
         INSERISCI_IN_TESTA(selezione, ELEMENTO(partecipanti, i))
         TOGLI(popolazione,ELEMENTO(popolazione,i))
      end if
   end while

   return selezione
end
```

In cui:

- *disordina(lista)*: prende in input una lista e ne sposta gli elementi riposizionandoli in ordine casuale, restituendo la nuova lista.
- *ordinaPerFitness(popolazione)*: prende in input una lista di orari e li ordina per fitness, dall'orario migliore a quello peggiore.
- *random()*: è una funzione che restituisce un numero casuale nell'intervallo [0,1].

4.5 Crossover

La finalità dell'operatore di crossover è quella di produrre, emulando il fenomeno biologico della riproduzione sessuata, degli individui *figli* attraverso la fusione del patrimonio genetico di un certo numero di genotipi. Questi genotipi (generalmente una coppia, come in questo caso) appartengono al sottoinsieme della popolazione "consegnato" dalla funzione di selezione.

La procedura proposta è:

```
Procedura incrocia(orario1, orario2)
```

```
begin
   allocate := Λ
   daAllocare := Λ

   for (a,c,p) ∈ orario1 do
      for (a',c',p') ∈ orario2 do
         if (a=a') ∧ (c=c') then
            if random() < 0.5 then
               lezione := (a,c,p)
            else
               lezione := (a,c,p')
            end if

            if allocabile(lezione, allocate) then
               allocate = INSERISCI_IN_TESTA(allocate,lezione)
            else
               daAllocare = INSERISCI_IN_TESTA(daAllocare,lezione)
            end if
         end if
      end for
   end for

   return assegnaPeriodi(allocate, daAllocare)
end
```

In cui:

- *random()*: è una funzione che restituisce un numero casuale nell'intervallo [0,1].
- *allocabile(lezione, allocate)*: è una funzione che prende in input una *lezione* candidata ad essere programmata ed una lista di lezioni già *allocate*. Si occupa di controllare che l'eventuale introduzione della nuova lezione non produca la rottura di alcun vincolo hard, e restituisce *true* se ciò è vero, *false* altrimenti.

Da notare come la funzione *assegnaPeriodi*, già vista precedentemente, agisca in questo caso come funzione di riparazione [8].

4.6 Mutazione

Un classico operatore di di mutazione per un algoritmo genetico agisce ciecamente su alcuni geni al fine di consentire l'esplorazione di quelle zone non ancora raggiunte dello spazio delle soluzioni.

La procedura proposta, invece, tenta di velocizzare la convergenza andando ad agire forzatamente sulle lezioni che stanno rompendo qualche vincolo, e che dunque sappiamo essere migliorabili. Questo meccanismo, che si potrebbe definire di tipo *Baldwiniano* [9], consiste in:

```
Procedura mutazioneIntelligente(orario, mutationSize)
begin
   allocate := orario
   daAllocare := Λ

   allocate := ordinaPerMigliorabilità(allocate)
   for i := 0 to ⌈mutationSize / 2⌉ do
      lezione := ELEMENTO(allocate,i)
      daAllocare = INSERISCI_IN_TESTA(daAllocare, lezione)
      allocate = TOGLI(allocate,lezione)
   end for

   allocate := disordina(allocate)
   for i := 0 to ⌊mutationSize / 2⌋ do
      lezione := ELEMENTO(allocate,i)
      daAllocare = INSERISCI_IN_TESTA(daAllocare, lezione)
      allocate = TOGLI(allocate,lezione)
   end for

   return assegnaPeriodi(allocate, daAllocare)
end
```

In cui:

- *ordinaPerMigliorabilità(lezioni)*: prende in input una lista di lezioni e restituisce una lista contenente le stesse lezioni ordinate sulla base di quanti vincoli rompono.
- *disordina(lezioni)*: prende in input una lista di lezioni, sposta gli elementi riposizionandoli in un ordine casuale e restituisce la nuova lista.

5 Benchmark

5.1 Un caso d'esempio

Come caso di esempio si vuole presentare una comune scuola secondaria di primo grado italiana (scuola media). A causa della difficoltà incontrata nel cercare di recuperare la descrizione esaustiva delle risorse di un istituto realmente esistente, è stata scelta la strada di compilare un'istanza fittizia, seppur plausibile. Il vantaggio ottenuto, in compenso, è stato quello di poterla pianificare con la certezza che una soluzione ottima potesse esistere, allocando tutte le risorse necessarie.

5.1.1 Istanza

Compatibilmente con la normativa vigente, la scuola prevederà il seguente piano settimanale di insegnamento:

piano settimanale

Discipline	Ore 1° anno	Ore 2° anno	Ore 3° anno
Italiano, storia e geografia	5+2+2	5+2+2	5+2+2
Matematica e scienze	4+2	4+2	4+2
Lingua inglese	3	3	3
Seconda lingua comunitaria	3	3	3
Tecnologia	2	2	2
Arte e immagine	2	2	2
Scienze motorie e sportive	2	2	2
Musica	2	2	2
Religione cattolica	1	1	1

da svolgersi in *5 periodi* di docenza (da 60 minuti) al giorno per *6 giorni*, dal lunedì al sabato.

La scuola presenta 4 sezioni: A, B, C, D; ognuna formata da 3 classi per un totale di *12 classi*, ciascuna con un numero variabile di alunni compreso tra 18 e 30.

Le sezioni A e B studieranno Francese come seconda lingua comunitaria, mentre le sezioni C e D studieranno Spagnolo.

Classi dell'istituto

Classe	Sezione	Numero alunni	Seconda lingua comunitaria
1ª	A	30	Francese
2ª	A	25	Francese
3ª	A	18	Francese
1ª	B	30	Francese
2ª	B	25	Francese
3ª	B	18	Francese
1ª	C	30	Spagnolo
2ª	C	25	Spagnolo
3ª	C	18	Spagnolo
1ª	D	30	Spagnolo
2ª	D	25	Spagnolo
3ª	D	18	Spagnolo

L'istituto ha a disposizione *12 aule* ordinarie con una capienza di 20 o 30 alunni ed una *palestra* con una capienza di 50 alunni.

Aule dell'istituto

Aula	Piano	Capienza	Tipo
101	1	30	aula ordinaria
102	1	20	aula ordinaria
103	1	30	aula ordinaria
104	1	30	aula ordinaria
105	1	30	aula ordinaria
106	1	30	aula ordinaria
107	1	30	aula ordinaria
108	1	30	aula ordinaria
109	1	30	aula ordinaria
110	1	30	aula ordinaria
201	2	30	aula ordinaria
202	2	30	aula ordinaria
203	2	30	aula ordinaria
204	2	30	aula ordinaria
205	2	30	aula ordinaria
206	2	30	aula ordinaria
207	2	20	aula ordinaria
208	2	20	aula ordinaria
209	2	30	aula ordinaria
210	2	20	aula ordinaria
301	3	30	aula ordinaria
302	3	30	aula ordinaria
palestra	1	50	palestra

Le lezioni di scienze motorie e sportive richiedono di essere svolte in palestra, mentre tutte le altre materie possono essere insegnate in un'aula ordinaria.

Vi sono *25 insegnanti* a comporre il corpo docente, ognuno dotato di un giorno libero (in cui il docente risulta *non disponibile*) ma disponibile ad insegnare in tutti gli altri *periodi* per tutti i restanti giorni (per comodità di lettura il livello di *disponibilità* è stato impostato a 0).

Essi sono:

insegnanti

Nome	Cognome	Materia	Sezioni	Ore	Giorno libero
Dante	Alighieri	Italiano	A	15	Lunedì
Gabriele	D'Annunzio	Italiano	B	15	Lunedì
Giacomo	Leopardi	Italiano	C	15	Martedì
Giovanni	Boccaccio	Italiano	D	15	Martedì
Cristoforo	Colombo	Storia e geografia	A	12	Lunedì
Amerigo	Vespucci	Storia e geografia	B	12	Martedì
Marco	Polo	Storia e geografia	C	12	Mercoledì
Umberto	Nobile	Storia e geografia	D	12	Mercoledì
Giuseppe Ludovico	Lagrangia	Matematica e scienze	A	18	Giovedì
Leonardo	Fibonacci	Matematica e scienze	B	18	Giovedì
Nicolò	Tartaglia	Matematica e scienze	C	18	Venerdì
Girolamo	Cardano	Matematica e scienze	D	18	Venerdì
William	Shakespeare	Inglese	A, B	18	Lunedì
Charles	Dickens	Inglese	C, D	18	Martedì
Gustave	Flaubert	Francese	A, B	18	Mercoledì
Jorge Luis	Borges	Spagnolo	C, D	18	Mercoledì
Leonardo	Da Vinci	Tecnologia	A, B	12	Giovedì
Alessandro	Volta	Tecnologia	C, D	12	Giovedì
Michelangelo	Buonarroti	Arte e immagine	A, B	12	Venerdì
Michelangelo	Merisi	Arte e immagine	C, D	12	Venerdì
Fausto	Coppi	Scienze motorie e sportive	A, B	12	Sabato
Pietro	Mennea	Scienze motorie e sportive	C, D	12	Sabato
Antonio	Vivaldi	Musica	A, B	12	Sabato
Giuseppe	Verdi	Musica	C, D	12	Sabato
Karol	Wojtyła	Religione cattolica	A, B, C, D	12	Sabato

Il numero totale di lezioni da pianificare risulta essere 360.

5.1.2 Soluzione ottima

Di seguito verrà esposta in dettaglio una soluzione ottima trovata dall'algoritmo:

Orario 1ª A

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Arte** *Michelangelo Buonarroti* 106	**Storia** *Cristoforo Colombo* 106	**Storia** *Cristoforo Colombo* 106	**Inglese** *William Shakespeare* 106	**Geografia** *Cristoforo Colombo* 106	**Inglese** *William Shakespeare* 106
2ª ora	**Francese** *Gustave Flaubert* 106	**Scienze motorie** *Fausto Coppi* palestra	**Matematica** *Giuseppe Lagrangia* 106	**Italiano** *Dante Alighieri* 106	**Italiano** *Dante Alighieri* 106	**Matematica** *Giuseppe Lagrangia* 106
3ª ora	**Tecnologia** *Leonardo Da Vinci* 106	**Matematica** *Giuseppe Lagrangia* 106	**Scienze motorie** *Fausto Coppi* palestra	**Arte** *Michelangelo Buonarroti* 106	**Francese** *Gustave Flaubert* 106	**Geografia** *Cristoforo Colombo* 106
4ª ora	**Matematica** *Giuseppe Lagrangia* 106	**Italiano** *Dante Alighieri* 106	**Tecnologia** *Leonardo Da Vinci* 106	**Musica** *Antonio Vivaldi* 106	**Inglese** *William Shakespeare* 106	**Scienze** *Giuseppe Lagrangia* 106
5ª ora	**Scienze** *Giuseppe Lagrangia* 106	**Francese** *Gustave Flaubert* 106	**Italiano** *Dante Alighieri* 106	**Religione** *Karol Wojtyła* 106	**Musica** *Antonio Vivaldi* 106	**Italiano** *Dante Alighieri* 106

Orario 2ª A

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Matematica** *Giuseppe Lagrangia* 101	**Scienze motorie** *Fausto Coppi* palestra	**Matematica** *Giuseppe Lagrangia* 101	**Geografia** *Cristoforo Colombo* 101	**Italiano** *Dante Alighieri* 101	**Italiano** *Dante Alighieri* 101
2ª ora	**Scienze** *Giuseppe Lagrangia* 101	**Italiano** *Dante Alighieri* 101	**Tecnologia** *Leonardo Da Vinci* 101	**Inglese** *William Shakespeare* 101	**Francese** *Gustave Flaubert* 101	**Tecnologia** *Leonardo Da Vinci* 101
3ª ora	**Religione** *Karol Wojtyła* 101	**Musica** *Antonio Vivaldi* 101	**Italiano** *Dante Alighieri* 101	**Francese** *Gustave Flaubert* 101	**Scienze** *Giuseppe Lagrangia* 101	**Inglese** *William Shakespeare* 101
4ª ora	**Arte** *Michelangelo Buonarroti* 101	**Storia** *Cristoforo Colombo* 101	**Inglese** *William Shakespeare* 101	**Storia** *Cristoforo Colombo* 101	**Geografia** *Cristoforo Colombo* 101	**Francese** *Gustave Flaubert* 101
5ª ora	**Musica** *Antonio Vivaldi* 101	**Arte** *Michelangelo Buonarroti* 101	**Scienze motorie** *Fausto Coppi* palestra	**Italiano** *Dante Alighieri* 101	**Matematica** *Giuseppe Lagrangia* 101	**Matematica** *Giuseppe Lagrangia* 101

Orario 3ª A

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Religione** *Karol Wojtyła* 210	**Matematica** *Giuseppe Lagrangia* *210*	**Italiano** *Dante Alighieri* 210	**Italiano** *Dante Alighieri* 210	**Scienze motorie** *Fausto Coppi* palestra	**Scienze** *Giuseppe Lagrangia* *210*
2ª ora	**Tecnologia** *Leonardo Da Vinci* 210	**Francese** *Gustave Flaubert* 210	**Inglese** *William Shakespeare* 210	**Scienze motorie** *Fausto Coppi* palestra	**Geografia** *Cristoforo Colombo* 210	**Italiano** *Dante Alighieri* 210
3ª ora	**Matematica** *Giuseppe Lagrangia* *210*	**Italiano** *Dante Alighieri* 210	**Storia** *Cristoforo Colombo* 210	**Storia** *Cristoforo Colombo* 210	**Tecnologia** *Leonardo Da Vinci* 210	**Matematica** *Giuseppe Lagrangia* *210*
4ª ora	**Musica** *Antonio Vivaldi* 210	**Inglese** *William Shakespeare* 210	**Geografia** *Cristoforo Colombo* 210	**Francese** *Gustave Flaubert* 210	**Francese** *Gustave Flaubert* 210	**Inglese** *William Shakespeare* 210
5ª ora	**Arte** *Michelangelo Buonarroti* 210	**Scienze** *Giuseppe Lagrangia* *210*	**Matematica** *Giuseppe Lagrangia* *210*	**Musica** *Antonio Vivaldi* 210	**Italiano** *Dante Alighieri* 210	**Arte** *Michelangelo Buonarroti* 210

Orario 1ª B

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Matematica** *Leonardo Fibonacci* *206*	**Musica** *Antonio Vivaldi* 206	**Scienze** *Leonardo Fibonacci* *206*	**Italiano** *Gabriele D'Annunzio* 206	**Scienze** *Leonardo Fibonacci* *206*	**Tecnologia** *Leonardo Da Vinci* 206
2ª ora	**Storia** *Amerigo Vespucci* 206	**Arte** *Michelangelo Buonarroti* 206	**Matematica** *Leonardo Fibonacci* *206*	**Storia** *Amerigo Vespucci* 206	**Inglese** *William Shakespeare* 206	**Geografia** *Amerigo Vespucci* 206
3ª ora	**Arte** *Michelangelo Buonarroti* 206	**Francese** *Gustave Flaubert* 206	**Italiano** *Gabriele D'Annunzio* 206	**Scienze motorie** *Fausto Coppi* palestra	**Matematica** *Leonardo Fibonacci* *206*	**Italiano** *Gabriele D'Annunzio* 206
4ª ora	**Francese** *Gustave Flaubert* 206	**Scienze motorie** *Fausto Coppi* palestra	**Musica** *Antonio Vivaldi* 206	**Inglese** *William Shakespeare* 206	**Italiano** *Gabriele D'Annunzio* 206	**Matematica** *Leonardo Fibonacci* *206*
5ª ora	**Tecnologia** *Leonardo Da Vinci* 206	**Italiano** *Gabriele D'Annunzio* 206	**Religione** *Karol Wojtyła* 206	**Francese** *Gustave Flaubert* 206	**Geografia** *Amerigo Vespucci* 206	**Inglese** *William Shakespeare* 206

Orario 2ª B

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Storia** *Amerigo Vespucci* 302	**Francese** *Gustave Flaubert* 302	**Inglese** *William Shakespeare* 302	**Musica** *Antonio Vivaldi* 302	**Inglese** *William Shakespeare* 302	**Italiano** *Gabriele D'Annunzio* 302
2ª ora	**Matematica** *Leonardo Fibonacci* *302*	**Matematica** *Leonardo Fibonacci* *302*	**Scienze motorie** *Fausto Coppi* palestra	**Italiano** *Gabriele D'Annunzio* 302	**Italiano** *Gabriele D'Annunzio* 302	**Arte** *Michelangelo Buonarroti* 302
3ª ora	**Scienze** *Leonardo Fibonacci* *302*	**Italiano** *Gabriele D'Annunzio* 302	**Tecnologia** *Leonardo Da Vinci* 302	**Geografia** *Amerigo Vespucci* 302	**Geografia** *Amerigo Vespucci* 302	**Matematica** *Leonardo Fibonacci* *302*
4ª ora	**Religione** *Karol Wojtyła* 302	**Scienze** *Leonardo Fibonacci* *302*	**Italiano** *Gabriele D'Annunzio* 302	**Scienze motorie** *Fausto Coppi* palestra	**Matematica** *Leonardo Fibonacci* *302*	**Tecnologia** *Leonardo Da Vinci* 302
5ª ora	**Francese** *Gustave Flaubert* 302	**Inglese** *William Shakespeare* 302	**Musica** *Antonio Vivaldi* 302	**Arte** *Michelangelo Buonarroti* 302	**Francese** *Gustave Flaubert* 302	**Storia** *Amerigo Vespucci* 302

Orario 3ª B

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Francese** *Gustave Flaubert* 208	**Inglese** *William Shakespeare* 208	**Italiano** *Gabriele D'Annunzio* 208	**Arte** *Michelangelo Buonarroti* 208	**Storia** *Amerigo Vespucci* 208	**Francese** *Gustave Flaubert* 208
2ª ora	**Religione** *Karol Wojtyła* 208	**Italiano** *Gabriele D'Annunzio* 208	**Geografia** *Amerigo Vespucci* 208	**Francese** *Gustave Flaubert* 208	**Musica** *Antonio Vivaldi* 208	**Matematica** *Leonardo Fibonacci* *208*
3ª ora	**Musica** *Antonio Vivaldi* 208	**Scienze motorie** *Fausto Coppi* palestra	**Inglese** *William Shakespeare* 208	**Italiano** *Gabriele D'Annunzio* 208	**Italiano** *Gabriele D'Annunzio* 208	**Arte** *Michelangelo Buonarroti* 208
4ª ora	**Scienze motorie** *Fausto Coppi* palestra	**Tecnologia** *Leonardo Da Vinci* 208	**Scienze** *Leonardo Fibonacci* *208*	**Geografia** *Amerigo Vespucci* 208	**Tecnologia** *Leonardo Da Vinci* 208	**Italiano** *Gabriele D'Annunzio* 208
5ª ora	**Matematica** *Leonardo Fibonacci* *208*	**Matematica** *Leonardo Fibonacci* *208*	**Matematica** *Leonardo Fibonacci* *208*	**Storia** *Amerigo Vespucci* 208	**Inglese** *William Shakespeare* 208	**Scienze** *Leonardo Fibonacci* *208*

Orario 1ª C

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Scienze motorie** *Pietro Mennea* palestra	**Spagnolo** *Jorge Luis Borges* 301	**Tecnologia** *Alessandro Volta* 301	**Scienze motorie** *Pietro Mennea* palestra	**Inglese** *Charles Dickens* 301	**Scienze** *Nicolò Tartaglia* 301
2ª ora	**Inglese** *Charles Dickens* 301	**Musica** *Giuseppe Verdi* 301	**Religione** *Karol Wojtyła* 301	**Matematica** *Nicolò Tartaglia* 301	**Storia** *Marco Polo* 301	**Inglese** *Charles Dickens* 301
3ª ora	**Matematica** *Nicolò Tartaglia* 301	**Matematica** *Nicolò Tartaglia* 301	**Italiano** *Giacomo Leopardi* 301	**Italiano** *Giacomo Leopardi* 301	**Italiano** *Giacomo Leopardi* 301	**Tecnologia** *Alessandro Volta* 301
4ª ora	**Italiano** *Giacomo Leopardi* 301	**Arte** *Michelangelo Merisi* 301	**Matematica** *Nicolò Tartaglia* 301	**Geografia** *Marco Polo* 301	**Spagnolo** *Jorge Luis Borges* 301	**Arte** *Michelangelo Merisi* 301
5ª ora	**Spagnolo** *Jorge Luis Borges* 301	**Storia** *Marco Polo* 301	**Musica** *Giuseppe Verdi* 301	**Scienze** *Nicolò Tartaglia* 301	**Geografia** *Marco Polo* 301	**Italiano** *Giacomo Leopardi* 301

Orario 2ª C

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Italiano** *Giacomo Leopardi* 110	**Storia** *Marco Polo* 110	**Inglese** *Charles Dickens* 110	**Scienze** *Nicolò Tartaglia* 110	**Spagnolo** *Jorge Luis Borges* 110	**Inglese** *Charles Dickens* 110
2ª ora	**Musica** *Giuseppe Verdi* 110	**Matematica** *Nicolò Tartaglia* 110	**Scienze** *Nicolò Tartaglia* 110	**Geografia** *Marco Polo* 110	**Italiano** *Giacomo Leopardi* 110	**Tecnologia** *Alessandro Volta* 110
3ª ora	**Tecnologia** *Alessandro Volta* 110	**Spagnolo** *Jorge Luis Borges* 110	**Matematica** *Nicolò Tartaglia* 110	**Inglese** *Charles Dickens* 110	**Storia** *Marco Polo* 110	**Italiano** *Giacomo Leopardi* 110
4ª ora	**Geografia** *Marco Polo* 110	**Religione** *Karol Wojtyła* 110	**Scienze motorie** *Pietro Mennea* palestra	**Spagnolo** *Jorge Luis Borges* 110	**Scienze motorie** *Pietro Mennea* palestra	**Matematica** *Nicolò Tartaglia* 110
5ª ora	**Matematica** *Nicolò Tartaglia* 110	**Arte** *Michelangelo Merisi* 110	**Italiano** *Giacomo Leopardi* 110	**Italiano** *Giacomo Leopardi* 110	**Musica** *Giuseppe Verdi* 110	**Arte** *Michelangelo Merisi* 110

Orario 3ª C

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Matematica** *Nicolò Tartaglia* 102	**Scienze** *Nicolò Tartaglia* 102	**Matematica** *Nicolò Tartaglia* 102	**Italiano** *Giacomo Leopardi* 102	**Geografia** *Marco Polo* 102	**Spagnolo** *Jorge Luis Borges* 102
2ª ora	**Italiano** *Giacomo Leopardi* 102	**Tecnologia** *Alessandro Volta* 102	**Tecnologia** *Alessandro Volta* 102	**Inglese** *Charles Dickens* 102	**Scienze motorie** *Pietro Mennea* palestra	**Scienze** *Nicolò Tartaglia* 102
3ª ora	**Spagnolo** *Jorge Luis Borges* 102	**Geografia** *Marco Polo* 102	**Inglese** *Charles Dickens* 102	**Matematica** *Nicolò Tartaglia* 102	**Musica** *Giuseppe Verdi* 102	**Matematica** *Nicolò Tartaglia* 102
4ª ora	**Musica** *Giuseppe Verdi* 102	**Storia** *Marco Polo* 102	**Italiano** *Giacomo Leopardi* 102	**Religione** *Karol Wojtyła* 102	**Italiano** *Giacomo Leopardi* 102	**Italiano** *Giacomo Leopardi* 102
5ª ora	**Arte** *Michelangelo Merisi* 102	**Scienze motorie** *Pietro Mennea* palestra	**Arte** *Michelangelo Merisi* 102	**Spagnolo** *Jorge Luis Borges* 102	**Inglese** *Charles Dickens* 102	**Storia** *Marco Polo* 102

Orario 1ª D

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Spagnolo** *Jorge Luis Borges* 203	**Musica** *Giuseppe Verdi* 203	**Arte** *Michelangelo Merisi* 203	**Scienze** *Girolamo Cardano* 203	**Religione** *Karol Wojtyła* 203	**Matematica** *Girolamo Cardano* 203
2ª ora	**Geografia** *Umberto Nobile* 203	**Spagnolo** *Jorge Luis Borges* 203	**Inglese** *Charles Dickens* 203	**Spagnolo** *Jorge Luis Borges* 203	**Tecnologia** *Alessandro Volta* 203	**Arte** *Michelangelo Merisi* 203
3ª ora	**Scienze motorie** *Pietro Mennea* palestra	**Matematica** *Girolamo Cardano* 203	**Matematica** *Girolamo Cardano* 203	**Matematica** *Girolamo Cardano* 203	**Storia** *Umberto Nobile* 203	**Scienze** *Girolamo Cardano* 203
4ª ora	**Italiano** *Giovanni Boccaccio* 203	**Storia** *Umberto Nobile* 203	**Musica** *Giuseppe Verdi* 203	**Italiano** *Giovanni Boccaccio* 203	**Italiano** *Giovanni Boccaccio* 203	**Italiano** *Giovanni Boccaccio* 203
5ª ora	**Inglese** *Charles Dickens* 203	**Tecnologia** *Alessandro Volta* 203	**Italiano** *Giovanni Boccaccio* 203	**Geografia** *Umberto Nobile* 203	**Scienze motorie** *Pietro Mennea* palestra	**Inglese** *Charles Dickens* 203

Orario 2ª D

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Italiano** *Giovanni Boccaccio* 103	**Geografia** *Umberto Nobile* 103	**Scienze motorie** *Pietro Mennea* palestra	**Musica** *Giuseppe Verdi* 103	**Italiano** *Giovanni Boccaccio* 103	**Italiano** *Giovanni Boccaccio* 103
2ª ora	**Spagnolo** *Jorge Luis Borges* 103	**Matematica** *Girolamo Cardano* 103	**Matematica** *Girolamo Cardano* 103	**Italiano** *Giovanni Boccaccio* 103	**Musica** *Giuseppe Verdi* 103	**Geografia** *Umberto Nobile* 103
3ª ora	**Matematica** *Girolamo Cardano* 103	**Tecnologia** *Alessandro Volta* 103	**Arte** *Michelangelo Merisi* 103	**Religione** *Karol Wojtyła* 103	**Inglese** *Charles Dickens* 103	**Arte** *Michelangelo Merisi* 103
4ª ora	**Inglese** *Charles Dickens* 103	**Spagnolo** *Jorge Luis Borges* 103	**Italiano** *Giovanni Boccaccio* 103	**Matematica** *Girolamo Cardano* 103	**Tecnologia** *Alessandro Volta* 103	**Spagnolo** *Jorge Luis Borges* 103
5ª ora	**Scienze** *Girolamo Cardano* 103	**Scienze** *Girolamo Cardano* 103	**Inglese** *Charles Dickens* 103	**Scienze motorie** *Pietro Mennea* palestra	**Storia** *Umberto Nobile* 103	**Storia** *Umberto Nobile* 103

Orario 3ª D

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì	Sabato
1ª ora	**Inglese** *Charles Dickens* 207	**Matematica** *Girolamo Cardano* 207	**Religione** *Karol Wojtyła* 207	**Inglese** *Charles Dickens* 207	**Geografia** *Umberto Nobile* 207	**Tecnologia** *Alessandro Volta* 207
2ª ora	**Arte** *Michelangelo Merisi* 207	**Storia** *Umberto Nobile* 207	**Italiano** *Giovanni Boccaccio* 207	**Arte** *Michelangelo Merisi* 207	**Spagnolo** *Jorge Luis Borges* 207	**Scienze** *Girolamo Cardano* 207
3ª ora	**Italiano** *Giovanni Boccaccio* 207	**Geografia** *Umberto Nobile* 207	**Tecnologia** *Alessandro Volta* 207	**Italiano** *Giovanni Boccaccio* 207	**Scienze motorie** *Pietro Mennea* palestra	**Italiano** *Giovanni Boccaccio* 207
4ª ora	**Spagnolo** *Jorge Luis Borges* 207	**Musica** *Giuseppe Verdi* 207	**Matematica** *Girolamo Cardano* 207	**Storia** *Umberto Nobile* 207	**Musica** *Giuseppe Verdi* 207	**Inglese** *Charles Dickens* 207
5ª ora	**Scienze motorie** *Pietro Mennea* palestra	**Spagnolo** *Jorge Luis Borges* 207	**Scienze** *Girolamo Cardano* 207	**Matematica** *Girolamo Cardano* 207	**Italiano** *Giovanni Boccaccio* 207	**Matematica** *Girolamo Cardano* 207

5.2 Parametri e impostazioni

L'algoritmo descritto nel capitolo precedente lasciava una certa libertà sia nella scelta dei valori da assegnare agli operatori genetici che nella scelta delle condizioni di terminazione, questo perché essi rappresentano una scelta molto delicata da compiere in funzione della specifica istanza del problema e delle risorse a disposizione.

Tale scelta costituisce di per sé un ulteriore problema di ottimizzazione, e sono note in letteratura molteplici implementazioni cosiddette *auto-adattive*. Per un ulteriore approfondimento si veda [10].

5.2.1 Parametri per gli operatori genetici

In questo caso, i parametri scelti sono frutto di tuning manuale. I parametri utilizzati per gli operatori genetici sono i seguenti:

parametri utilizzati

Parametro	Valore
popSize	20
tournamentSize	10
pCross	0.9
pMutate	0.8
mutationSize	10

Aumentare la dimensione della popolazione (*popSize*) portava ad una significativa riduzione del numero di generazioni necessarie a trovare una soluzione ottima, ma senza un reale vantaggio in termini di tempo di esecuzione, probabilmente a causa di un utilizzo non ottimale della parallelizzazione (l'argomento sarà trattato nell'ultimo capitolo). Diminuirla, invece, riduceva nettamente l'efficacia dell'algoritmo.

Agire sulla probabilità d'accoppiamento degli individui migliori (*tournamentSize* e *pCross*) portava ad una convergenza prematura verso una soluzione non ottima in caso di probabilità troppo alta, e ad un andamento praticamente casuale nel caso fosse troppo bassa.

Per quanto riguarda la mutazione, invece, si sono riscontrati buoni risultati sia utilizzando mutazioni significative (*mutationSize*) poco frequentemente (*pMutate*) che facendo l'opposto, ovvero piccole mutazioni eseguite molto frequentemente. Nel primo caso vi era una convergenza molto veloce verso popolazioni con valori migliori, ma non è stata mai raggiunta la soluzione ottima. Nel secondo, invece, si è ottenuta una convergenza più lenta ma costante fino alla soluzione ottima.

5.2.2 Condizione di terminazione

Avendo a che fare con un'istanza del problema creata ad-hoc, vi è la certezza che esista una soluzione ottima. Per questo motivo si è potuto decidere di non limitare il numero di iterazioni ad alcun tempo massimo di esecuzione né numero limite di generazioni, ma proseguendo fino al raggiungimento di tale soluzione.

5.3 Risultati ottenuti

Per l'istanza del problema appena presentata e con i parametri illustrati nel paragrafo precedente, il software è riuscito a generare una soluzione ottima in un tempo medio di circa *20 minuti (1196918 millisecondi)* e in mediamente *99 generazioni*.

5.3.1 Riepilogo

Si riporta di seguito la tabella riassuntiva di 20 esecuzioni consecutive dell'algoritmo sull'istanza presentata.

riepilogo esecuzioni

Esecuzione	Minuti di esecuzione	Numero di generazioni
1	13,61	87
2	16,87	97
3	16,27	94
4	14,72	90
5	17,61	93
6	11,45	75
7	33,23	124
8	26,44	107
9	24,89	122
10	17,43	96
11	23,10	110
12	11,50	77
13	9,77	68
14	11,54	63
15	14,76	78
16	17,26	99
17	24,63	107
18	35,87	148
19	33,23	122
20	24,79	118
Media	19,95	98,75
Deviazione standard	7,88	21,45957031

Il tempo ed il numero di generazioni richieste sono così distribuiti:

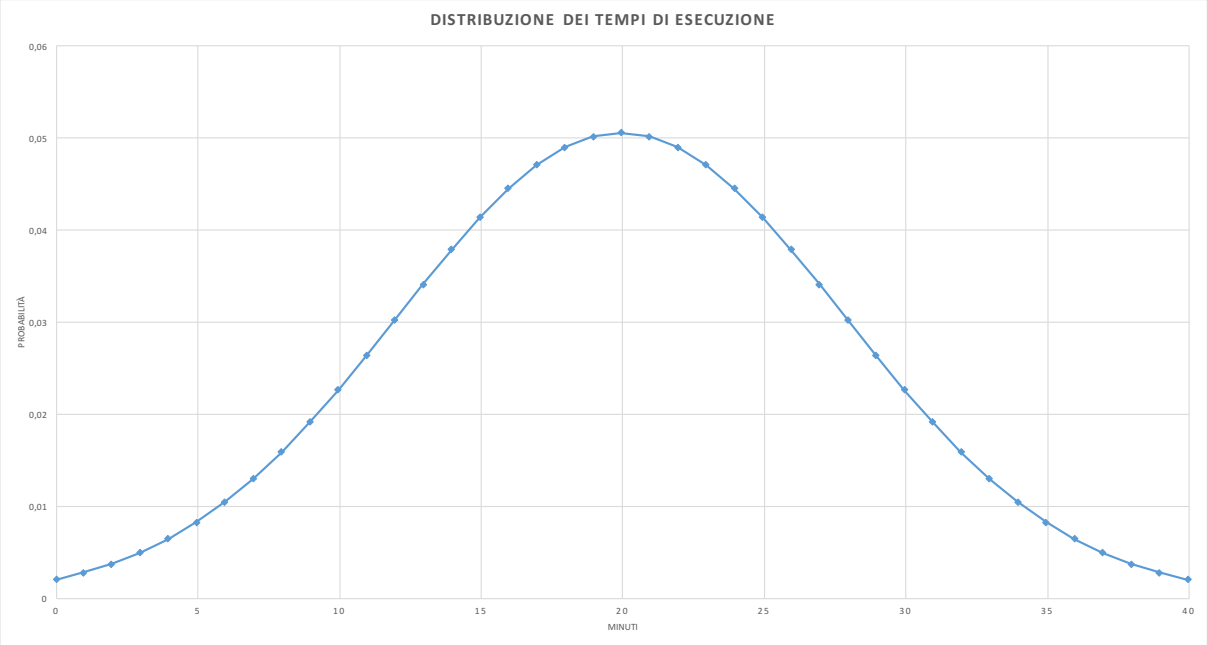

FIGURA 5.1 DISTRIBUZIONE DEI TEMPI DI ESECUZIONE

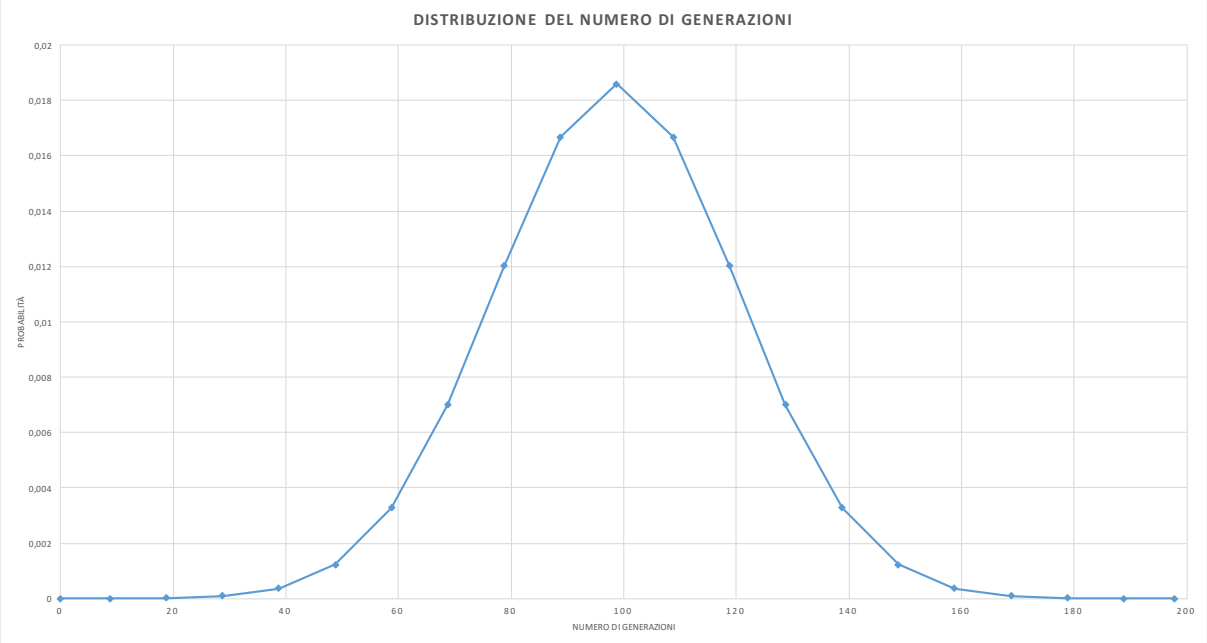

FIGURA 5.2 DISTRIBUZIONE DEL NUMERO DI GENERAZIONI

5.3.2 Andamento medio

Nel grafico seguente viene mostrato l'andamento medio evidenziando la deviazione standard.

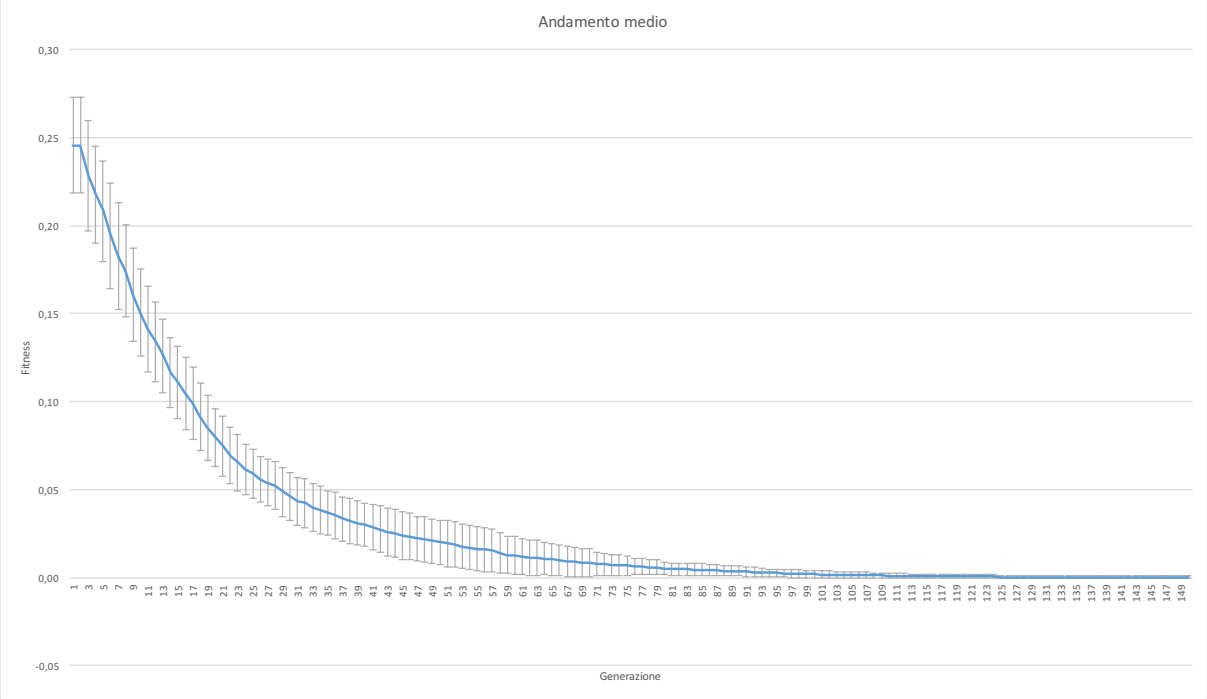

FIGURA 5.3 ANDAMENTO MEDIO DELL'ALGORITMO

Per ogni esecuzione, comunque vi è una sempre una chiara convergenza verso la soluzione ottima.

5.3.3 Un confronto con il Simulated Annealing

Utilizzando un'altra strategia di ottimizzazione, ovvero il *Simulated Annealing*, e limitando il tempo di esecuzione a 20 minuti, non è mai stata raggiunta una soluzione ottima. Di seguito, un grafico che riporta l'andamento medio:

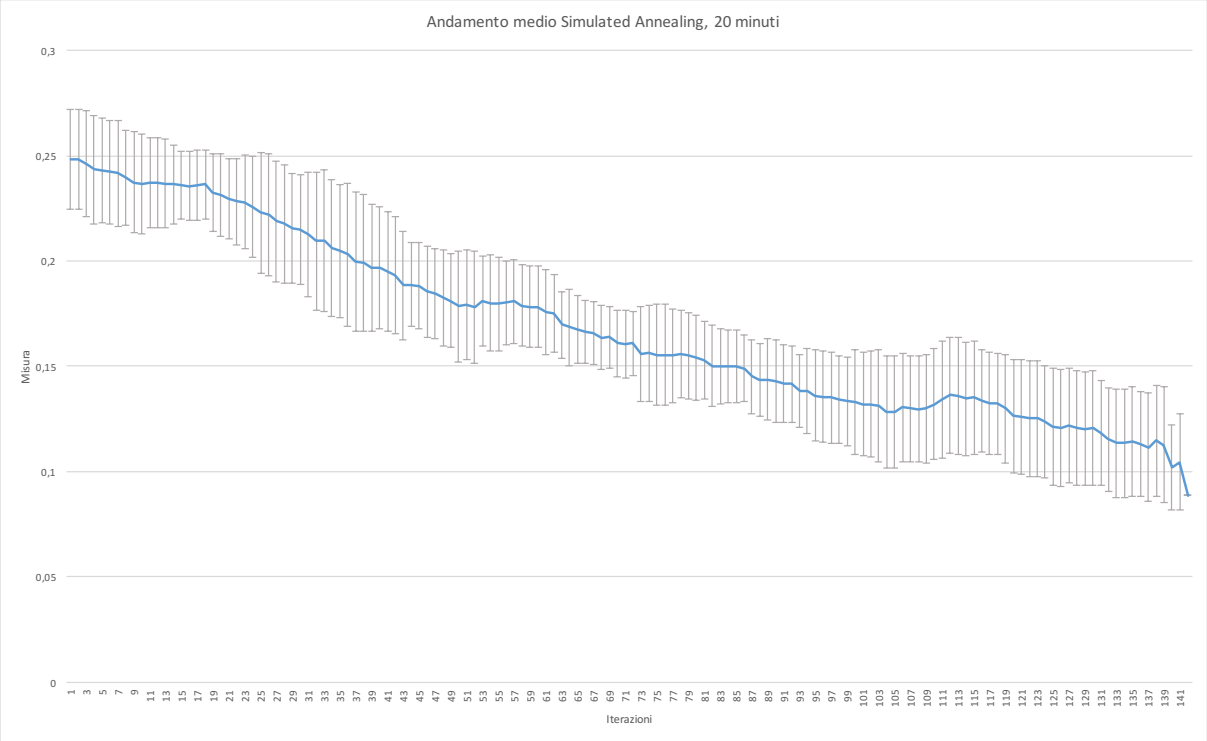

FIGURA 5.4 ANDAMENTO MEDIO DEL SIMULATED ANNEALING

Anche questo algoritmo si rivela efficace, ma i tempi richiesti per la convergenza risultano maggiori: non è stato possibile trovare una soluzione ottima neanche lasciando il software in esecuzione per diverse ore.

6 Sviluppi futuri

6.1 Miglioramento dell'efficienza

Sono sicuramente molti gli aspetti migliorabili sia dal punto di vista algoritmico che implementativo.

Un primo passo consisterà nel refactoring del codice volto ad alleggerirlo da tutti quegli elementi inizialmente necessari al tracking ed all'analisi dei risultati.

Successivamente, si prevede di intervenire sul codice del motore di ricerca delle soluzioni e fare in modo che esso gestisca in maniera intelligente la computazione su più processi, sfruttando la natura parallelizzabile degli algoritmi evolutivi. L'obiettivo è quello di fare in modo che tutte le operazioni non necessariamente sequenziali vengano effettuate in contemporanea dividendo il carico tra i processori disponibili. Pare che in alcuni casi si sia riuscito ad ottenere uno speed-up quasi lineare su determinati algoritmi genetici [11].

Per limitare ulteriormente l'interdipendenza delle procedure si potrebbe passare ad un modello ad isole, che consiste nel dividere la popolazione in un insieme di sottopopolazioni e consentendogli di evolversi in maniera autonoma, prevedendo poi solo saltuariamente la migrazione casuale di alcuni individui da un'isola all'altra.

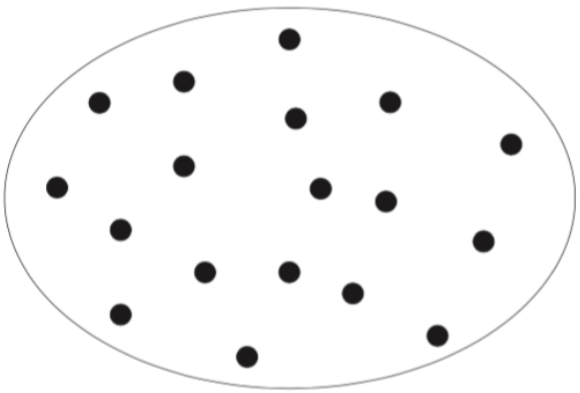

FIGURA 6.1 UNIVERSO PANMITTICO, UTILIZZATO NELL'IMPLEMENTAZIONE ATTUALE

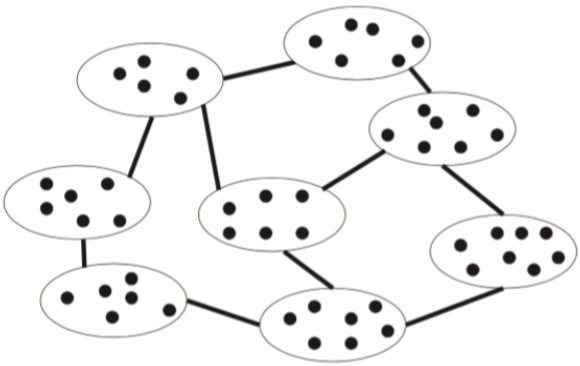

FIGURA 6.2 UNIVERSO AD ISOLE, CHE POTREBBE ESSERE UTILIZZATO PER MIGLIORARE LA PARALLELIZZAZIONE

Altre strade percorribili al fine di migliorare l'efficienza sono quelle di prevedere che i parametri degli operatori genetici si adattino autonomamente in base all'istanza corrente (come discusso nel paragrafo 5.2) ed evitare la rivisitazione di soluzioni già esplorate [12].

6.2 Funzionalità aggiuntive

Bisognerà realizzare un'interfaccia grafica completa per l'utente finale. Questo passo sarà di importanza cruciale, dato che potrebbe determinare anche da solo il successo o meno del software.

Il vero vantaggio commerciale, però, sarà costituito dalla declinazione su realtà concrete, ottenibile attraverso aggiunte mirate al modello dei dati ed ai vincoli trattati al fine di introdurre delle funzionalità aggiuntive studiate ad-hoc per il cliente. Alcuni esempi potrebbero essere:

- Gestione di più edifici e limitazione degli spostamenti.
- Pianificazione del servizio mensa e delle pause pranzo.
- Riduzione delle ore di inattività (ore buche) a cavallo tra due lezioni.
- Gestione delle lezioni condivise tra più classi o co-impartite da più docenti in compresenza.
- Omogeneizzazione del carico di lavoro giornaliero degli studenti, ottenibile associando un peso ad ogni lezione in base alla materia trattata e distribuendo equamente la somma di tali pesi nell'arco della settimana. Si potrebbe anche cercare di ottenere un carico di lavoro decrescente nell'arco delle singole giornate, cosa che si è rivelata opportuna dal punto di vista didattico.
- Pre-assegnazione manuale di determinate lezioni.
- Previsione dell'indisponibilità di alcuni locali in certi periodi di tempo.

Bibliografia

[1] Decreto Legislativo 14 settembre 2011, n. 167 “Testo unico dell’apprendistato, a norma dell’articolo 1, comma 30, della legge 24 dicembre 2007, n. 247”, articolo 5.

[2] Willemen, Robertus J., School timetable construction : algorithms and complexity, Eindhoven : Technische Universiteit Eindhoven, 2002. Proefschrift. ISBN 90-386-1011-4

[3] Di Stefano, C., “Algoritmi evolutivi per la risoluzione del problema dell’orario, Università degli Studi di Milano”, 1998.

[4] Tettamanzi, A., Tomassini, M., *Soft Computing: integrating evolutionary, neural, and fuzzy systems*, Springer-Verlag Berlin Heidelberg, 2001 ISBN 3-540-42204-8

[5] Mitchell, Melanie (1996). An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press. ISBN 9780585030944.

[6] Poli, R., Langdon, W. B., McPhee, N. F. (2008), *A Field Guide to Genetic Programming*, Lulu.com, disponibile gratuitamente via internet ISBN 978-1-4092-0073-4

[7] Whitley, Darrell (1994). "A genetic algorithm tutorial". Statistics and Computing 4 (2): 65–85. doi:10.1007/BF00175354.

[8] Orvosh, D., Davis, L (1994) “Using a genetic algorithm to optimize problems with feasibility constraints”; doi:10.1109/ICEC.1994.350001

[9] Chen, X. S.; Ong, Y. S.; Lim, M. H.; Tan, K. C. (2011). "A Multi-Facet Survey on Memetic Computation". IEEE Transactions on Evolutionary Computation 15 (5): 591–607. doi:10.1109/tevc.2011.2132725

[10] Eiben, A.E., Hinterding, R., Michalewicz, Z.; *Parameter control in evolutionary algorithms*; Evolutionary Computation, IEEE Transactions on (Volume:3 , Issue: 2) [124-141] doi:10.1109/4235.771166.

[11] David Andre, John R. Koza; "*A parallel implementation of genetic programming that achieves super-linear performance"*; Information Sciences: an International Journal. Volume 106 Issue 3-4, May, 1998; [201-218]; doi: 10.1016/S0020-0255(97)10011-1

[12] Shiu Yin Yuen, Chi Kin Chow; (2008); "*A Genetic Algorithm That Adaptively Mutates and Never Revisits*"; Evolutionary Computation, IEEE Transactions on (Volume:13 , Issue: 2) [454-472]; doi: 10.1109/TEVC. 2008.2003008