ESCUELA SUPERIOR POLIT ´

ECNICA DEL LITORAL



´

INSTITUTO DE CIENCIAS MATEM ATICAS ESCUELA DE GRADUADOS

´

TESIS DE GRADUACI ON

´

PREVIO A LA OBTENCI ON DEL T ´ITULO DE ´

MAG ´ISTER EN CONTROL DE OPERACIONES Y GESTI ON LOG ´ISTICA

TEMA

´

DISENO E IMPLEMENTACI ˜ON DE UNA HEUR ´ISTICA PARA RESOLVER EL ´

PROBLEMA DE CALENDARIZACI ON DE HORARIOS PARA UNIVERSIDADES

AUTOR JOSE XAVIER CABEZAS GARC ´IA

Guayaquil-Ecuador 2009

Dedicatoria

Ami esposaLenay amishijosDanielayXavierEduar­do. Los amo con todas mis fuerzas.

Agradecimientos

ADiospordarmelafortalezapara alcanzar mis metas, a mispadrespor su apoyoincondicional, alMat.Fernando Sandoyapor su acertadadirecci´on en estainvestigaci´on ya mis compa˜nerosde trabajoMat.JohnniBustamante e Ing. Erwin Delgado por su amistad y por sus ´utiles consejos profesionales.

Declaraci´on Expresa

La responsabilidad del contenido de esta Tesis de Postgrado, me corresponde exclu­sivamente; y el patrimonio intelectual del mismo al ICM(Instituto de Ciencias Matem´aticas) de la ESPOL(Escuela Superior Polit´ecnica Del Litoral).

(Reglamento de Graduaci´on de la ESPOL)

Jos´e Xavier Cabezas Garc´ıa

´

TRIBUNAL DE GRADUACI ON

M.Sc. F´elix Ram´ırez Cruz PRESIDENTE DEL TRIBUNAL

M.Sc. Fernando Sandoya Sanchez M.Sc. Johnni Bustamante Romero DIRECTOR DE TESIS VOCAL

´

Indice general

1. El Problema de Calendarizaci´on de Horarios para Universidades 12

1.1. Introducci´onalaCalendarizaci´on . . . . . . . . . . . . . . . . . . . . 12

1.2. Enfoques de soluciones para Calendarizaci´on . . . . . . . . . . . . . . 15

1.2.1. Deﬁnici´on de Heur´ıstica y Metaheur´ıstica . . . . . . . . . . . . 15

1.2.2. Resumen de algunos m´etodos de soluci´on . . . . . . . . . . . . 16

1.3. Descripci´ondelCB-CTT . . . . . . . . . . . . . . . . . . . . . . . . . 17

1.3.1. Entidades que participan en el CB-CTT . . . . . . . . . . . . 17

1.3.2. RestriccionesdelCB-CTT .................... 18

1.3.3. UnmodelodeProgramaci´onLineal . . . . . . . . . . . . . . . 19

2. El Algoritmo Gen´etico para el CB-CTT 21

2.1. Introducci´on a los Algoritmos Gen´eticos . . . . . . . . . . . . . . . . 21

2.1.1. Relaci´onconlaevoluci´on. . . . . . . . . . . . . . . . . . . . . 21

2.1.2. DeﬁnicionesFormalesenlosAG’s . . . . . . . . . . . . . . . . 23

2.1.3. Procedimientos Principales en los AG’s . . . . . . . . . . . . . 24

2.2. Algoritmo Gen´etico para Calendarizaci´on . . . . . . . . . . . . . . . . 26

2.2.1. Elementos fundamentales en la Calendarizaci´on . . . . . . . . 26

2.2.2. Operadores gen´eticos en la Calendarizaci´on . . . . . . . . . . . 28

3. Implementaci´on computacional de un AG para el CB-CTT 30

®

3.1. El entorno MatlabR. . . . . . . . . . . . . . . . . . . . . . . . . . . 30

®

3.1.1. Caracter´ısticas de MatlabR. . . . . . . . . . . . . . . . . . . 30

3.1.2. Recursos computacionales empleados . . . . . . . . . . . . . . 31

3.2. Datosbaseparaimplementaci´on. . . . . . . . . . . . . . . . . . . . . 32

3.2.1. Formatodedatosdeentrada. . . . . . . . . . . . . . . . . . . 32

3.2.2. Formulaciones. . . . . . . . . . . . . . . . . . . . . . . . . . . 33

3.2.3. Instancias ............................. 34

®

3.2.4. Matrices de entrada para el AG en MatlabR. . . . . . . . . . 35

®

3.3. Dise˜no del AG para el CB-CTT en MatlabR. . . . . . . . . . . . . . 37

3.3.1. Representaci´on cromos´omica del calendario . . . . . . . . . . . 37

3.3.2. Algoritmoparacrearuncromosoma . . . . . . . . . . . . . . 39

3.3.3. Programa de generaci´on de la Poblaci´on Inicial . . . . . . . . 39

3.3.4. Formadelafunci´ondeevaluaci´on . . . . . . . . . . . . . . . . 39

3.3.5. Evaluaci´ondeunapoblacion. . . . . . . . . . . . . . . . . . . 40

3.3.6. Elprocesodeselecci´on ...................... 41

3.3.7. Dise˜nodelprocedimientodecruce . . . . . . . . . . . . . . . . 42

3.3.8. Funci´ondecruceparalaPoblaci´on . . . . . . . . . . . . . . . 42

3.3.9. Mutaci´ondeuncromosoma ................... 43

3.3.10. Funci´on de mutaci´on para la poblaci´on . . . . . . . . . . . . . 44

3.3.11.CriteriodeparadadelAG .................... 44

3.3.12.AG completo ........................... 45

4. Experimentos computacionales 46

4.1. Calibraci´ondepar´ametros ........................ 46

4.2. Soluciones al problema de Calendarizaci´on . . . . . . . . . . . . . . . 52

4.2.1. BasededatosTOY . . . . . . . . . . . . . . . . . . . . . . . . 52

4.2.2. Unejemplocontrescurr´ıculums. . . . . . . . . . . . . . . . . 53

5. Conclusiones y Recomendaciones 56

5.1. Conclusiones. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56

5.2. Recomendaciones ............................. 57

A. Programas adicionales 58

B. Calendarios adicionales 73

C. Datos de entrada para ejemplo de tres curr´ıculums 74

´

Indice de ﬁguras

1.1. EjemplodeCalendarioEduacional(Timetable) . . . . . . . . . . . . 13

2.1. Representaci´ondeuncromosoma .................... 22

2.2. Ejemplo de cruce de dos puntos con genes binarios . . . . . . . . . . . 24

2.3. Mutaci´onenunarepresentaci´onbinaria. . . . . . . . . . . . . . . . . 24

2.4. Ejemplo del m´etodo de selecci´on Ruleta de la Fortuna. . . . . . . . . 25

2.5. Representaci´on de un cromosoma para el problema de calendarizaci´on. 27

2.6. Ejemplo de Poblaci´on Inicial para Calendarizaci´on. .......... 27

2.7. Ilustraci´on del operador de cruce queevitacolisiones. . . . . . . . . . 29

2.8. Ilustraci´on del operador de mutaci´on. . . . . . . . . . . . . . . . . . . 29

®

3.1. Entorno MatlabR. ............................ 31

3.2. Instancias de la Universidad de Udine vista desde la p´agina Web de SaTT. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

®

3.3. Salida de datos de entrada en interfaz MatlabR. . . . . . . . . . . . 37

3.4. Representaci´onmatricialdeuncromosoma(calendario). . . . . . . . 38

3.5. Ejemplo de aplicaci´on de la funci´on ﬁtness . . . . . . . . . . . . . . . 41

3.6. Ejemplo del vector ﬁtnessp para una poblaci´on de tama˜no 5 . . . . . 41

3.7. Ejemplo de un proceso de cruce . . . . . . . . . . . . . . . . . . . . . 43

3.8. Ejemplo de un proceso de mutacion . . . . . . . . . . . . . . . . . . . 44

4.1. Tama˜no de Poblaci´on vs Fitness, variaci´on de poblaci´on . . . . . . . . 48

4.2. Tama˜no de Poblaci´on vs N´umero de Iteraciones, variaci´on de poblaci´on 48

4.3. Tama˜no de Poblaci´on vs Tiempo de Ejecuci´on, variaci´on de poblaci´on 49

4.4. Probabilidad de Cruce vs Fitness, variaci´on de probabilidad de cruce 50

4.5. ProbabilidaddeCrucevsN´umerodeIteraciones, variaci´ondeprobabi­lidad de cruce . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50

4.6. ProbabilidaddeCrucevsTiemposdeEjecuci´on,variaci´ondeprobabi­lidad de cruce . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50

4.7. Probabilidad de Mutaci´on vs Fitness, variaci´on de probabilidad de mutaci´on . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51

4.8. Probabilidad de Mutaci´on vs N´umero de Iteraciones, variaci´on de probabilidad de mutaci´on . . . . . . . . . . . . . . . . . . . . . . . . 51

4.9. ProbabilidaddeMutaci´onvsTiemposdeEjecuci´on,variaci´ondepro­babilidad de mutaci´on . . . . . . . . . . . . . . . . . . . . . . . . . . 52

4.10.Calendariopara elCurr´ıculum1, Toy . . . . . . . . . . . . . . . . . 52

4.11.Calendariopara elCurr´ıculum2, Toy . . . . . . . . . . . . . . . . . 53

4.12. Aplicaci´on de la funci´on ﬁtness a la base de datos Toy . . . . . . . . 53

4.13.Calendario enformato matriz,parala soluci´onde Toy . . . . . . . . 54

4.14.CalendarioparaelCurr´ıculum1,ejemplo2 . . . . . . . . . . . . . . . 54

4.15.CalendarioparaelCurr´ıculum2,ejemplo2 . . . . . . . . . . . . . . . 55

4.16.CalendarioparaelCurr´ıculum3,ejemplo2 . . . . . . . . . . . . . . . 55

4.17. Aplicaci´on de la funci´on ﬁtness a la base de datos del ejemplo 2 . . . 55

B.1. Calendario para el Curr´ıculum 1, SaTT, Toy . . . . . . . . . . . . . 73

B.2. Calendario para el Curr´ıculum 2, SaTT, Toy . . . . . . . . . . . . . 73

´

Indice de tablas

1.1. Clasiﬁcaci´on de Problemas de Calendarizaci´on Educacional . . . . . . 13

2.1. Resumen de penalizaciones por incumplimiento de restricciones. . . . 28

3.1. Recursos para implementaci´on computacional del AG . . . . . . . . . 31

3.2. Descripci´on de formulaciones de problemas de calendarizaci´on . . . . 34

3.3. Contenido del archivo toy.txt . . . . . . . . . . . . . . . . . . . . . . . 36

3.4. Nombres y c´odigos de los archivos txt . . . . . . . . . . . . . . . . . . 37

3.5. Procedimientos de la funci´on ﬁtness . . . . . . . . . . . . . . . . . . 40

4.1. Experimentocomputacional(Tama˜nodePoblaci´on) . . . . . . . . . . 48

4.2. Experimentocomputacional(ProbabilidaddeCruce) . . . . . . . . . 49

4.3. Experimentocomputacional(ProbabilidaddeMutaci´on) . . . . . . . 51

C.1. Base de datos para ejemplo de tres curr´ıculums . . . . . . . . . . . . 75

´

Indice de funciones y

®

procedimientos en MatlabR

1. “readﬁles.m”: Procedimiento para leer archivos externos txt . . . . 35

2. “gpob.m”: Funci´on que genera la poblaci´on inicial . . . . . . . . . . 39

3. “ﬁtness.m”: Funci´on de evaluaci´on para un cromosoma . . . . . . . 40

4. “evaluapoblacion.m”: Devuelve un vector con el ﬁtness de cada cromosoma de una poblaci´on . . . . . . . . . . . . . . . . . . . . . . . 41

5. “seleccion.m”: Selecciona individuos para nueva generaci´on . . . . . 42

6. “criterioparada.m”: Criterio de parada caso 1, proporci´on de ele­mentos con igual ﬁtness . . . . . . . . . . . . . . . . . . . . . . . . . 45

7. “agcbctt.m”:AG completo. . . . . . . . . . . . . . . . . . . . . . . 45

8. “pruebas.m”: Genera continuas corridas del AG . . . . . . . . . . . 47

9. “gcrom.m”: Genera un cromosoma . . . . . . . . . . . . . . . . . . 59

10. “roomycap.m”: Ubica un aula y su capacidad . . . . . . . . . . . . 60

11. “mejoraroom.m”: Mejora un cromosoma asignando nuevas aulas . 60

12. “generalecyprof.m”: Ubica una lectura y un profesor . . . . . . . . 61

13. “roomoccupancy.m”: Procedimiento para Aulas ocupadas . . . . . 62

14. “conﬂicts.m”: Procedimiento para Conﬂictos . . . . . . . . . . . . . 63

15. “uconstraints.m”: Procedimiento para Disponibilidad . . . . . . . . 63

16. “roomcapacity.m”: Procedimiento para Capacidad de aulas . . . . 64

17. “minwdays.m”: Procedimiento para D´ıas de trabajo m´ınimo . . . . 64

18. “ccompactness.m”: Procedimiento para Curr´ıculum compacto . . . 65

19. “rstability.m”: Procedimiento para Estabilizaci´on de aulas . . . . . 66

20. “cruce.m” Parte I: Proceso de cruce de un par de cromosomas . . 67

21. “cruce.m” Parte II: Proceso de cruce de un par de cromosomas . . 68

22. “cruce.m” Parte III: Proceso de cruce de un par de cromosomas . 69

23. “crucepoblacion.m”:Seleccionaa unpadrey una madreparacru­zarlos y elige quien formar´a parte de la nueva generaci´on . . . . . . . 70

24. “mutacion.m”: Ubica un gen en un periodo disponible. . . . . . . . 71

25. “mutacionpoblacion.m”: Deﬁne si un cromosoma entra alproceso de mutaci´on . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72

Cap´ıtulo 1

El Problema de Calendarizaci´on de Horarios para Universidades

1.1. Introducci´on a la Calendarizaci´on

ElProblemadeCalendarizaci´on TTP(por sus siglas eningl´es Timetabling)est´aden­tro de un conjunto de problemas m´as amplio denominados simplemente de Progra­maci´on (Scheduling)[11]ysepuededeﬁnirdeforma muy simple.A continuaci´on un par de deﬁniciones que aparecen en la literatura:

Deﬁnici´on 1.1.1 (Calendarizaci´on,Zhipeng LuandJin-Kao Hao[7]). Asignar un n´umero de eventos, cada uno con ciertas caracter´ısticas, a un n´umero limitado de recursos sujeto a restricciones.

LanceD.Chambers[2] deﬁnetambi´en al TTP de una forma m´as general:

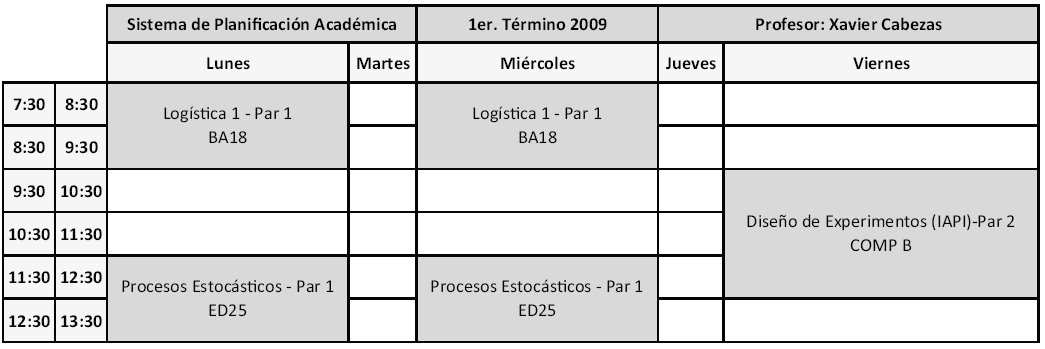
Deﬁnici´on 1.1.2 (Calendarizaci´on, Chambers L.). El Timetabling (Calendariza­ci´on) se puede deﬁnir como aquello que describe, d´onde y cu´ando las personas y los recursos deben estar en un instante dado.

Para poder interpretar lo que en la deﬁnici´on 1.1.1 signiﬁcan los t´erminos: eventos, recursosy restricciones, sedebeestudiarlas caracter´ısticasde unproblemaparticu­lar. Para este trabajo, se considera una clase de TTP’s denominada Educacional, que consiste enlaprogramaci´on de clases de un conjuntode materias con un n´ume­ro dado de aulas y periodos de tiempo disponiblesque satisfacen varias restricciones [1], ver Figura 1.1. Las restricciones pueden ser clasiﬁcadas enduras y suaves. Las restricciones de tipo duras son utilizadas para encontrar soluciones factibles al pro­blema y no pueden ser infringidas, por otro lado las restricciones suaves pueden relajarsedetalformadebuscarsolucionescercanasal ´ancotas(in­

optimo que ser´feriores o superiores) para buscar nuevas alternativas de soluci´on.

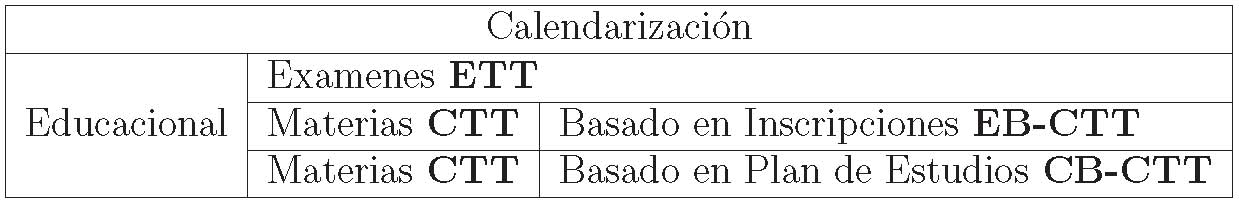
El caso Educacional es solo una de las variaciones de los TTP’s, tambi´en existen otrasaplicacionesen ´areasmuydistintas,talescomotransporte(programaci´onde salidasdebuses),TV(CalendariodeProgramasdeTV) odeportes(programaci´on decalendariosdejuegos),unejemplode ´este ´ultimotipodeproblemaseencuentra en[15],donde se muestra un m´etodode soluci´onque utiliza una aproximaci´on con programaci´onlineal entera(ILP, por las siglas en ingl´es de Integer Linear Progra­mming)para la determinaci´on de horarios de la Liga Italiana de F´utbol.

Figura 1.1: Ejemplo de Calendario Eduacional(Timetable)



Para el caso Educacional se pueden clasiﬁcar en dos grupos: Calendarizaci´on pa­ra Ex´amenes (ETT, Exam Timetabling)y Calendarizaci´on para Materias (Clases Regulares)(CTT, Course Timetabling), este ´ultimo a su vez puede ser subdividido en Calendarizaci´on por Materias Basado en Inscripciones (EB-CTT, Enrollment-Based Course Timetabling)yCalendarizaci´on por Materias Basado en Plan de Es­tudios (CB-CTT, Curriculum-Based Course Timetabling), ver Tabla 1.1.

Tabla 1.1: Clasiﬁcaci´on de Problemas de Calendarizaci´on Educacional



Cuando seintenta resolver el TTP,puedenpresentarse una seriede conﬂictos entre lasrestriccionesqueseplantean,pero ´estasparael casodel EB-CTT se deber´an a ladecisi´onquetomenlosestudiantes sobrelas materiasquedeseantomar,un cruce podr´ıa ocurrir si la elecci´on no ha sido buena; por otro lado los problemas que se podr´anpresentar en el CB-CTT resultar´ande unaprogramaci´on notanbuenadel plan de estudios por parte de la instituci´on educativa.

Los conﬂictos ocurren cuando una programaci´on requiera que cualquiera de los re­cursos est´e endoslugares al mismo tiempo.La restriccionesde equipo var´ıan eviden­temente entre instituciones, entre ex´amenes y programaciones de clases regulares. En realidad tododependedel tama˜nodelainstituci´on odeldepartamentoque ofre­ce sus cursos y del tiempo requerido para producir la soluci´on.

Otra de las cosas complicadas en los problemas de Calendarizaci´on es el lograr satisfacer las restricciones de las diferentes personas que tienen inter´es sobre los resultados obtenidos.Losprincipales stakeholders1 en esteproblema con susprinci­

1Stakeholders esunt´erminoanglosaj´onutilizadopara referirse apersonasquetieneninter´es en un proceso determinado.

pales objetivos y necesidades son:

1. La Administraci´on: La cual deﬁne el m´ınimo est´andar requerido para una “Tabla de Tiempo”, como por ejemplo, el n´umero de ex´amenes que un estu­diante deber´ıa tomar en un d´ıa espec´ıﬁco.
2. Los Departamentos: En algunas ocasiones, los departamentos dentro de las instituciones o institutos suelen tener demandas espec´ıﬁcas sobre aulas de clases o laboratorios.
3. Estudiantes: Es muy complicado obtener una tabla de tiempo que satisfaga todas las necesidades de cada uno de los estudiantes. Algunos estudiantes preﬁeren no tener clases regulares muy tarde y otros en cambio lo preﬁeren as´ı para tener oportunidad de trabajar. Por otro lado, una exigencia muy com´un, esque entre cada examen consecutivo exista undescanso(break),que les premita un peque˜no espacio entre prueba y prueba o entre clase y clase.

LosprofesoresBurke,Jackson,Kingstony Weare[10] mencionanque entrelasdife­rentesformasdelosproblemasde Calendarizaci´on sepueden encontrar restricciones comunes que los relacionan unos con otros:

Asignaci´on de recursos: Cuando se requieren diferentes recursos depen­diendo de las necesidades de los cursos dictados o ex´amenes a tomar.

Asignaci´on de tiempo: Esta restricci´on se utiliza cuando sedesead´ıaspara los cuales los profesores est´an disponibles, o para pre asignar tiempo a una clase o examen particular.

Restricci´on de tiempo entre clases o ex´amenes: Estas restricciones se reﬁeren a las condiciones de tiempo entre sesiones, por ejemplo, un conjunto de ex´amenespodr´ıanquerer tomarse al mismo tiempo, o un conjuntode clases podr´ıan querer ser dadas en un orden determinado.

Sesiones dispersas: Las sesiones deber´ıan ser dispersas en el tiempo. Este tipo de restricciones son las mencionadas anteriormente y que son impuestas por la administraci´on, por ejemplo, podr´ıa requerirse que los estudiantes no tomen m´as de un examen en un d´ıa y si lo hacen, deber´ıan ser muy pocos, para que hayan intervalos adecuados de descanso.

Sesiones coherentes: Estas restricciones son las que consideran condiciones particulares de los participantes. Las restricciones de los estudiantes forman partede estejuego,porquedeber´ıan exigirintervalosdedescanso entre ex´a­menes si tienenque tomar m´asde unaprueba en und´ıa, olosprofesoresquiz´as preﬁeran dar clases regulares en d´ıas seguidos.

Capacidad de salas: Se reﬁeren a la capacidad f´ısica de los cursos donde se impartir´anlas clases regulares(incluyendolaboratorios) odonde setomar´an ex´amenes.

Continuidad: Son cualquiergrupoderestriccionesqueasegurencaracter´ısti­cas de una “tabla de tiempo” (horario de clases)de estudiantes, por ejemplo, un mismo curso (grupo de estudiantes que toman las mismas materias o el mismo examen) deber´ıa etar programado en una misma aula.

1.2. Enfoques de soluciones para Calendarizaci´on

1.2.1. Deﬁnici´on de Heur´ıstica y Metaheur´ıstica

Los problemas de Calendarizaci´on Educacional son aplicados a centros educativos detodoslosniveles:Escuelas,Colegios, Universidades.Estetrabajotrataderesolver calendarizacionespara Universidades las cualestienen caracter´ısticaspropiasincluso diferentes entre instituciones del mismo tipo, por ejemplo: Porcentaje de profesores a tiempo completo, aulas disponibles, n´umero de laboratorios de computaci´on, etc.

El esfuerzo(computacionaly humano) dequien oquienesplaniﬁcanloshorariosde clases en cualquier universidad ha hecho que el problema de calendarizaci´on para este caso particular sea considerado uno de los m´as relevantes en la optimizaci´on combinatoria. En muchos de los casos las soluciones encontradas de forma manual suelen dejar muchas de las restricciones del problema sin cumplir, y el descontento de los actores que de una u otra forma est´an relacionados con la construcci´on del horario ﬁnal, se hace notar de diferentes maneras.

Un camino para resolver el problema es la Programaci´on Lineal Entera(ILP) [3], que considera un conjunto de ecuaciones que representan las restricciones na­turales de la elaboraci´on de horarios, adem´as de una funci´on objetivo (FO) que puede representar comolohizoMolina[5] “el n´umero de cruces de horarios en la soluci´on” yquedebeserminimizada conlaesperanzaquetomeel valorde cero(0). Formulaciones de programaci´on lineal del problema y sus restricciones, se pue­den encontrar a lo largo de la literatura, sin embargo muchas de ellas no contienen algunas restricciones de fuerte inter´es y que se esperan se cumplan en la planiﬁca­ci´on. Muchas veces, esto es debido a que algunas restricciones suelen ser dif´ıciles de formular y limitan el problemas a instancias que no son aplicables a algunos de los casos reales. Aqu´ı es donde se consideran alternativas de soluci´on Heur´ısticas o Metaheur´ısticas como Algoritmos Gen´eticos [19], B´usqueda Tab´u [7], Re­cocido Simulado, entre otros, que encuentran soluciones aproximadas, pero que con un buen dise˜no pueden estar muy cerca del ´

optimo.

La palabra Heur´ıstica proviene del griego Heuriskein que signiﬁca “encontrar”, lo cual no parece ser una palabra adecuada para describir estos m´etodos, porque m´as que “encontrar” “busca”.

Deﬁnici´on1.2.1 (Heur´ısticas). Algoritmos simples, a menudobasados en el sentido com´un,quesesuponeofrecer´anunabuenasoluci´on(aunqueno necesariamentela ´optima) a problemas espec´ıﬁcos considerados dif´ıciles, de un modo f´acil y r´apido.

La principal diferencia entre heur´ısticas y metaheur´ısticas es que las primeras se dise˜nan para problemas particulares y solo para estos, y las segundas pueden uti­lizarse para resolver una gran variedad de problemas combinatorios adaptando su esquemageneral alproblemaparticular cambiando algunaspartesde su estructura.

Los Algoritmos Gen´eticos,porejemplo,hansido ´utilespararesolverla Planiﬁcaci´on de Proyectos conRecursos Restringidos [12], as´ı comoparaproblemasde Coloraci´on de Grafos como se menciona enD´ıaz[4].

Deﬁnici´on 1.2.2 (Metaheur´ısticas). Algoritmos basados en el sentido com´un o en procesosdeotras ´areasdelacienciaodelanaturaleza,quesirvenpararesolvermu­chosproblemasconsideradosdif´´

ıciles, encontrando soluciones aproximadas al optimo en un tiempo peque˜no.

1.2.2. Resumen de algunos m´etodos de soluci´on

Referencias espec´ıﬁcas sobre los tres m´etodos presentados a continuaci´on pueden encontrarse enD´ıaz[4].

AlgoritmosGen´eticos(GAs): Estos algor´ıtmos est´aninspirados enlateor´ıa de la evoluci´on de Darwin, Simulando el proceso de selecci´on de la naturale­za, con la esperanza de que as´ı se consigan ´exitos similares en relaci´on con la capacidad de adaptaci´on de un amplio n´umero de ambientes diferentes. La informaci´on hereditaria en los organismos es pasada a trav´es de los cromoso­mas que contienen la informaci´on de todos estos factores, es decir, los genes, los cuales a su vez est´an compuestos por un determinado n´umero de valores. Varios organismos se agrupan formando una poblaci´on, y aquellos que me­jor se adaptan son aquellos que tienen mayor probabilidades de sobrevivir y reproducirse. Algunos de los sobrevivientes son seleccionados para crear nue­vos organismos. Adem´as los genes de un cromosoma pueden sufrir cambios produciendo mutaciones en los organismos.

B´usqueda Tab´u (Tabu Search): Es un tipo de b´usqueda por entornos, el cual gu´ıa un procedimiento de b´usqueda local para explorar el espacio de soluciones m´as all´a del ´optimo local. La b´usqueda Tab´u selecciona de una maneraagresivael mejordelosmovimientosposiblesacadapaso. Alcontrario queenlab´usquedalocalquesiempresemueveal mejordesuentornoyﬁnaliza con la llegada a un ´optimo local, la b´usqueda Tab´u permite moverse a una soluci´onde su entorno o vecindadque no seatanbuena comola actual,detal manera que pueda tener oportunidad de salir de ´optimos locales y continuar estrat´egicamente la b´usqueda de soluciones a´un mejores. Para evitar ciclos clasiﬁca un determinado n´umero de los m´as recientes movi­mientos como “movimientos Tab´u”. Por lo tanto el escape de los ´optimos locales se produce de forma sistem´atica y no aleatoria. En otras palabras la B´usqueda Tab´u mantiene una memoria de eventos. La ﬁlosof´ıa de esta t´ecnica esla creenciadequela elecci´onde una mala estrategia sistem´aticadeb´usqueda es mejor que una buena elegida al azar.

Recocido Simulado(Simulated Annealing): Es un m´etodoheur´ısticoque guarda relaci´on con un campo muy diferente al de la optimizaci´on, la termo­din´amica.En cadaiteraci´on una vecindad esgenerada(Un horario factible se modiﬁca ligeramente de forma aleatoria para crear uno nuevo tambi´en facti­ble). Este vecino es aceptado como el actual horario si se considera que tiene baja penalidad. Si por otra parte, este mismo vecino tiene una alta penali­zaci´on, esta podr´ıa ser aceptada como la actual soluci´on, es decir como un calendario(horario) acorde a unaprobabilidad relacionada con unpar´ametro de controldenominadotemperatura.Latemperatura,yporlotantolaproba­bilidaddeque vecinos con altapenalidad sean aceptados, vadisminuyendo en cadaiteraci´on o m´as usualementedespu´esde un n´umerodeiteraciones(este n´umero puede ser constante o puede incrementarse de acuerdo a como dismi­nuye la temperatura). La relaci´on de este m´etodo con la termodin´amica est´a en el hecho de que el proceso se asemeja al enfriamiento en el recocido de metales.

Otras metodolog´ıasde soluci´on son muy novedosas, como el ElAlgoritmodeB´usque­da Dispersa (SSA por sus siglas en ingl´es Scatter Search Algorithm)[13],GRASP (Greedy Randomized Adaptive Search Procedure)y Redes Neuronales que suelen considerar restricciones poco comunes en relaci´on a los modelos est´andares.

1.3. Descripci´on del CB-CTT

1.3.1. Entidades que participan en el CB-CTT

Como se mencionaDiGaspero[1], al momentodehacer una revisi´ondelaliteratura del CB-CTT se pueden encontrar una serie de documentos que presentan nuevas formulaciones pero que no siempre consideran las previamente deﬁnidas por otros autores, lo cual es debido a que no es posible escribir una que contenga todos los casos posibles que pueden presentarse en la vida real, cada instituci´on educativa posee caracter´ısticasdiferentesquehacenque elproblema se vuelva muyparticular.

Con el ﬁn de tener una descripci´on general del CB-CTTP se ha tomado como referencialapropuestaporDiGaspero[17] que contienelas restricciones m´aspo­pulares(durasy suaves) delproblema.

Primero se comienza con deﬁnir las entidades que participan en el problema:

D´ıas(days), ranurasdetiempo(timeslots) yperiodos(periods): Se tiene un n´umero de d´ıas ala semanadisponiblepara ense˜nar(teaching days), los cuales se reparten en un n´umero ﬁjo de ranuras de tiempo iguales en todos los d´ıas. Un periodo es unpar ordenadodelaforma(d´ıa,ranura de tiempo). El n´umero total de periodos a programar son el producto de d´ıas por ranuras de tiempo.

Materias(courses) y profesores(teachers):Cada materia consiste en un n´umero ﬁjode lecturas (una hora clasede una materia, vienedelingl´es lecture, que aunque se traduce como conferencia simplemente se dir´a lectura)que se programar´an en distintos periodos, donde asisten un n´umero determinado de estudiantes y es impartido por un profesor. Cada materia deber´ıa impartirse en un m´ınimo n´umero de d´ıas a la semana, considerando adem´as que existen periodos en los cuales las materias no pueden ser programadas por alguna raz´on deﬁnida.

Aulas(rooms): Cada aula tiene una capacidad deﬁnida como el n´umero de lugares disponibles para los estudiantes (por ejemplo pupitres).

Curr´ıculum (curricula): Se deﬁne como un conjunto de materias tal que cualquier par de ellas en el grupo tienen estudiantes en com´un.

1.3.2. Restricciones del CB-CTT

Luego de cada restricci´on que se menciona a continuaci´on, se deﬁne cuando ocurre una violaci´on a ´esta,conelprop´ositodesaberquetanlejosseest´adeunasoluci´on perfectamente factible.

Restricciones Duras(HARD constraints)

Sesionesde clases(lectures): Todas las sesiones de clases de las materias deben programarse y asignarse a diferentes periodos. Si la sesi´on de clase no est´a programada, se considera una violaci´on.

AulasOcupadas(room occupancy): Dos sesiones de clases nopueden ser programas enla mismaaula en el mismo periodo.Se cuenta como una violaci´on adicional cualquier sesi´on de clase extra en una misma aula y periodo.

Conﬂictos (conﬂicts): Sesiones de clases de las materias del mismo cu­rriculum o dictadas por un mismo profesor deben programarse en diferentes periodos. Una violaci´on ocurre si dos sesiones de clases est´an en conﬂicto.

Disponibilidad(availabilities): Siun profesor de unamateria no est´adispo­nible para dictarla en un periodo dado, entonces ninguna sesi´on de clase de esta materia debe ubicarse en ese periodo. Cada sesi´on de clase en un periodo no disponible para esa materia se cuenta como una violaci´on.

RestriccionesSuaves(SOFT constraints)

CapacidaddeAulas(RoomCapacity):Para cadasesi´onde clase el n´ume­ro de estudiantes que asisten a las materias debe ser menor o igual al n´umero de lugares disponibles de todas las aulas que acogen a las sesiones de clases. Se cuenta como una violaci´on a cada alumno por encima de la capacidad del aula.

D´ıas de Trabajo M´ınimo (Minimum Working Days): Las sesiones de clases de cada materia deben serdictadas en un n´umero m´ınimode d´ıas.Cada

d´ıa por debajo del m´ınimo se consideran 5 puntos de penalidad.

Curriculum Compacto(Curriculum Compactnees): Sesiones de clases que pertenecen a un mismo curriculum deber´an estarjuntas unas con otras (periodos consecutivos). Para uncurriculum dado se considera una violaci´on cada vez que una sesi´on de clase no sea adyacente a otra en el mismo d´ıa.

Estabilizaci´on de Aulas (Room Stability): Todas las sesiones de clases de una materia deben ser dictadas en una sola aula.

En[1] seconsideranalastres ´ultimasrestriciconessuaves(D´ıas de Trabajo M´ınimo, Curriculum Compacto y Estabilizaci´on de Aulas)como componentes opcionales de costos, incluyendo adem´as:

Carga M´ınima y M´axima (Student Min Max Load): Para cada cu­rr´ıculum, el n´umero de sesiones de clases diarias deben estar dentro de un rango dado. Se considera una violaci´on a cada sesi´on de clase por encima o por debajo del rango impuesto.

DistanciadeTraslado(Travel Distance): Todoslos estudiantes deben te­ner el tiempo suﬁciente para trasladarse de un aula a otra, las cuales podr´ıan estar incluso en diferentes ediﬁcios, entre dos sesiones de clases. En esta res­tricci´on se cuenta como una violaci´on cada vezque sepresente un movimiento instant´aneo endos sesionesde clases en aulas localizadas endiferentes ediﬁcios en dos periodos adyacentes en un mismo d´ıa.

Idoneidad de Aulas (Room Suitability): Cuando una materia requiere equipos especiales (computadoras, proyector, etc.) en un aula espec´ıﬁca, se dice que ´esta no es id´onea. Cada sesi´on de clase de una materia programada en un aula no id´onea se considera como una violaci´on.

Sesiones de clases dobles (Double Lectures): Cuando se requiere que las sesiones de clases de una materia programadas en un mismo d´ıa est´en en ranuras de tiempo adyacentes y en la misma aula. Cada sesi´on de clase no agrupada se cuenta como una violaci´on a la restricci´on.

1.3.3. Un modelo de Programaci´on Lineal

Primero es necesario establecer los par´ametros y variables que intervienen en el modelo lineal, para luego deﬁnir las restriciones.

Par´ametros

m = N´umero de aulas n = N´umero materias p = N´umero de profesores q = N´umero de periodos [a,b]= Intervalo de materias correspondiente a un curr´ıculum [c,d]= Intervalo de periodos correspondiente a un d´ıa de la semana Ki = Capacidad del aula i. ∀i ∈{1,...,m} Cj = N´umero de estudiantes en la materia j. ∀j ∈{1,...,n}

Variables

1 Si el aula i es asignada a la materia j con el profesor k en el periodo l

Xijkl =

0 Si no ∀i ∈{1,...,m}, ∀j ∈{1,...,n}, ∀k ∈{1,...,p}y ∀l ∈{1,...,q}

Restricciones

1. Toda sala i debe ser asignada a un solo periodo.

np

LL

Xijkl ≤ 1 ∀i,l

j=1 k=1

2. Toda materia j tiene Pj periodos semanales.

mp q

LLL

Xijkl = Pj ∀j

i=1 k=1 l=1

3. Toda materia j debe ser dictada por un solo profesor k, en una sola aula i y en un solo periodo l.

mp

LL

Xijkl ≤ 1 ∀j,l

i=1 k=1

4. Todo profesor k debe dictar no m´as de una materia j en una ´unica aula i y en un solo periodo l.

mn

LL

Xijkl ≤ 1 ∀k,l

i=1 j=1

5. Nodebenprogramarse materias de un mismocurriculum en un mismo periodo.

pb m

LLL

Xijkl ≤ 1 ∀l

k=1 j=ai=1

6. La cantidad de estudiantes de la materia j debe ser menor o igual a la capa­cidad de la sala i.

Xijkl ≤ 1+(Ki −Cj) ∀i,j,k,l

La funci´on objetivo, puede ser formulada de m´as de una forma, un ejemplo puede encontrarse enMolina[5],por este motivo sehadejado su an´alisisparala siguiente secci´on. Hay que notar, como ya se ha discutido, que es muy complicado formular algunas restriciones,y como sepuede ver, este modelopuede mejorarse al incluirlas, sobre todo algunas de tipo suaves.

Cap´ıtulo 2

El Algoritmo Gen´etico para el CB-CTT

2.1. Introducci´on a los Algoritmos Gen´eticos

En el presente trabajo se ha optado por un m´etodo de soluci´on Metaheur´ıstico ba­sado en la evoluci´on de las soluciones (factibles o no) encontradas en diferentes iteraciones de un algoritmo al que se denomina Gen´etico, en parte por la experien­cia de trabajo con este tipo de modelos a trav´es de la preparaci´on de posgrado y en parte por la gran cantidad de bibliograf´ıa disponible sobre el tema, as´ı como el hechodeque eslabasede otros m´etodosque actualmente est´an enplenodesarrollo y cuya profundizaci´on por medio de la investigaci´on en aplicaciones a problemas complejos, pueden dar nuevas luces para otros procedimientos del mismo tipo.

Acontinuaci´on sedar´a unadescripci´ondelaformageneraldelos AlgoritmosGen´eti­cos y luego se dar´a un enfoque particular al problema de Calendarizaci´on por Ma­terias Basado en Plan de Estudios CB-CTT.

2.1.1. Relaci´on con la evoluci´on

John Holland ensulibro Adaptation in Natural and Aritiﬁcial Systems del a˜no1975

[8] fue el primero en utilizar el t´ermino Algortimo Gen´etico (al que abreviaremos AG), y que como su nombre lo indica, son procedimientos sistem´aticos basados en la seleci´on natural de los seres vivos y el paso de informaci´on gen´etica generaci´on a generaci´on. Holland bas´o su trabajo en la evoluci´on de las especies, propuesta por Darwin. Esta contribuci´on de Holland, ha servido de inspiraci´on para crear un campo de investigaci´on que se ha desarrollado mucho m´as all´a del trabajo original de los AG.

Deﬁnici´on 2.1.1 (Algoritmo Gen´etico[4]). Un algoritmo Gen´etico es una estruc­turade controlque organiza odirige un conjuntodetransformacionesy operaciones dise˜nadas para simular los procesos de evoluci´on.

Los primeros algoritmos de Holland eran simples, pero populares gracias a que pod´ıan resolver poblemas que al menos en esa ´epoca eran considerados dif´ıciles.

Los AG han evolucionado gracias a la contribuci´on de muchos autores a partir de eseentonces,inclusoel mismo Hollandhaincorporadoprocedimientosm´ascomple­josyloshamostradoenpublicacionesenlosa˜nosnoventas. Algunasmodiﬁcaciones han resultado en variantes h´ıbridas que mezclan conceptos de otras heur´ısticas y metaheur´ısticas conocidas.

L´ogicamente los t´erminos utilizados por los AG en sus implementaciones compu­tacionales guardan relaci´on con los de la evoluci´on natural y por este motivo es necesario describir algunas caracter´ısticas de este proceso, m´as a´un si lo que se re­quiere es un proceso de evoluci´on simulado,por medio de un algoritmo matem´atico implementado computacionalmente[4](p´ag 70).

Todas los cambios ocurridos a lo largo del cambio generacional ocurren en una unidad llamada cromosoma, que identiﬁca a cada miembro de una poblaci´on de in­dividuos,delos cuales nosinteresalaprobabilidadque sus caracter´ısticasgen´eticas sobrevivan en el futuro. La idea es que al transcurrir el tiempo en la poblaci´on solo queden los m´as fuertes, aquellos que producen buenas soluciones al problema que se plantea. Los Genes es la estructura m´as simple que forma un cromosoma, cada miembro de una poblaci´on tiene el mismo n´umero de genes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | · · · | j | · · · | N | |

Figura2.1:Representaci´onde un cromosoma. N representa el n´umerototaldegenes.

Losprimeros algoritmosAGrepresentaban alos genes por mediode valoresbinarios 1−0, sin embargo esto no es aplicable en la mayor´ıa de los problemas pr´acticos.

Laprincipalcaracter´ısticadelproceso evolutivo esla supervivencia,la elecci´on natu­raldel m´asfuerte,loque an´alogamentequieredecir en unproblemade optimizaci´on unindividuoqueprocure un mejor valorde unafunci´on objetivo(FO)a minimizar (o m´aximizar, de acuerdo al contexto de trabajo), es decir un individuo puede ser visto como una soluci´on posible que puede mejorar o empeorar a FO. Esto lleva a considerar una funci´on que permita medir que tan bueno o malo es un individuo respecto a otro. El cruce de los miembros de una poblaci´on, producen hijos con caracter´ısticasqueheredande suspadres,donde cadapareja se eligeprincipalmente por su fortaleza dentro del grupo al que pertenecen. La forma de cruzar genes de un cromosoma es un tema de discusi´on amplio, debido a que cada problema puede interpretarse y representarse de m´as de una manera.

Otro situaci´on que se vive en la naturaleza, es que de generaci´on en generaci´on siempreseproducencambiosestructuraleslos cuales se esperanque seanparabien, sin embargo con alguna probabilidad un descendiente pudiera traer deformaciones gen´eticas que lo hagan o bien vulnerable al medio ambiente o bien lo conviertan en una entidad superior, a esto se lo suele llamar mutaci´on.

Muchos autores han implementado AG para resolver problemas combinatorios de extremadiﬁcultad, comolohicieronMilenaKarova,VassilSmarkovyStoyanPenev [9],dondedesarrollanprocedimientosdecruce ymutaci´on para elconocidoproblema del Traveling Salesman Problem TSP y Maroto junto a Javier Alcaraz en [12], para un problema de Planiﬁcaci´on de Proyectos con Recursos Restringidos que se mencion´o en el cap´ıtulo anterior.

2.1.2. Deﬁniciones Formales en los AG’s

Es importante formalizar algunas deﬁniciones.

Problema de optimizaci´on combinatoria

Se supondr´a que se tiene un espacio de b´usqueda de soluciones discreto al que lla­maremos χ, y una funci´on f : χ → R, el problema radica en buscar soluciones en χ para minimar f, es decir: m´ınx∈χ f (o m´ax sabiendo que m´ınf = −m´ax(−f)), donde x es un vector de variables de decisi´on y f es la funci´on objetivo. A esta estructura se la conoce con el nombre de problema de optimizaci´on combinatoria o discreta.

Elvector x es representadopor una cadena s delargol constitu´ıdode s´ımbolosde un alfabeto Al gracias al mapping c : Al → χ. Como es posible que algunos elementos en el conjunto de llegada de c no sean soluciones factibles, podemos pensar que en realidad se necesita un subconjuntode Al alquellamaremos S, entonces S ⊆Al.Los elementos del string corresponden a los genes. A esto se lo llama mapping genotipo­fenotipo.

Deﬁnici´on 2.1.2 (Genotipo, Colin Reeves [6]). Representaci´on codiﬁcada de las variables.

Deﬁnici´on 2.1.3 (Fenotipo,ColinReeves[6]). Conjunto de variables en s´ı mismas.

El problema de optimizaci´on combinatoria puede escribirse entonces como:

m´ın g(s)

s∈S

donde g(s)= f(c(s)).

Representaci´on de variables, cruce y mutaci´on

Como se mencion´o enla secci´on anterior, la representaci´onde variables en los AG’s fue en principio utilizando cadenas s binarias, es decir A = {0,1}. En los AG’s se utiliza una Poblaci´on de cadenas las cuales se denominan cromosomas en el mismo contextodelas secci´on2.1.1.Larecombinaci´onde cadenas esloquehemosllamado cruce. Un ejemplo muy simple de cruce con representaci´on binaria es cuando se realizagenerando n´umeros aleatorios(2por ejemplo) queindicandesdedonde un hijo o hija mantienen los genes del padre (o de lamadre). En elejemplo siguiente, se ﬁjan los puntos 4 y 8 de cromosomas de tama˜no 10, en el primer caso, el hijo toma los genes del padre desde el gen 1 hasta el gen 4 y desde el gen 8 al 10, los genes intermedios corresponden a los de la madre, el caso contrario ocurre con la hija, ver Figura 2.2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 1 |  | |  |  | 2 | | |  | |  | | N´umeros Aleatorios | | | |
| 1 | 0 | 1 | 0 | 0 | | 0 | 1 | 0 | | | 0 | | 1 | | Padre | | | |
| 0 | 1 | 0 | 0 | 0 | | 1 | 1 | 1 | | | 0 | | 1 | | Madre | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 | | 1 | 1 | 0 | | | 0 | | 1 | | Hijo | |
| 0 | 1 | 0 | 0 | 0 | | 0 | 1 | 1 | | | 0 | | 1 | | Hija | |

Figura 2.2: Ejemplo de cruce de dos puntos con genes binarios

Un cambio en los genes se denomina mutaci´on. En el caso binario una mutaci´on es cambiar de gen de valor 1 a valor 0 y viceversa, ver Figura 2.3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 | | 0 | 1 | 0 | | | 0 | | 1 | | Cromosoma original | | |
| 1 | 0 | 1 | 0 | 0 | | 1 | 1 | 0 | | | 0 | | 1 | | Cromosoma mutado | | |

Figura 2.3: Mutaci´on en una representaci´on binaria

Labusquedadela soluci´on se consigue mediantela evaluaci´ondela funci´on objetivo f para cada cadena s de la poblaci´on. A la funci´on de evaluaci´on se la conoce con el nombre de ﬁtness y el procedimiento requiere que la cadena s con mayor valor de ﬁtness pueda seridentiﬁcado,para asignarle mayorprobabilidadde reproducirse.

2.1.3. Procedimientos Principales en los AG’s Poblaci´on Inicial

Antesde comenzar abuscar soluciones a unproblemaparticular, es necesario contar una un conjuntode soluciones(Poblaci´on Inicial) el cual pueda evolucionar gene­raci´on trasgeneraci´on.La Poblaci´on Inicial puede ser construidadeforma aleatoria

o mediante alg´unprocedimiento sistem´atico.Es muyimportante considerarque este conjunto debe contener suﬁciente informaci´on para que su transformaci´on procure una soluci´on ﬁnal ´optimo, una poblaci´

optima o al menos muy cerca del ´on muy homog´enea podr´ıa tener como consecuencia una convergecia hacia una soluci´on no necesariamente buena. Deber´ıa adem´as tener un tama˜no adecuado para evitar un bajo rendimiento del algoritmo, por otro lado una tama˜no de la poblaci´on muy grande podr´ıa llevarnos a un tiempo de proceso fuera de los l´ımites de lo esperado.

Selecci´on para una Nueva Generaci´on

Cuando ya se tiene una poblaci´on inicial diversa y de calidad se debe escoger los candidatos aquienes se aplicar´anlosprocedimientosde cruce y/omutaci´on, es claro que se requiere que estas soluciones deben ser individuos fuertes, lo que simula la elecci´on natural de las especies. Existen varios m´etodos para seleccionar soluciones entrelosque se encuentrandosdelas m´as conocidas: Torneo y Rueda de la Fortuna (Roulette Wheel).

En el m´etodo de selecci´on por Torneo se eligen aleatoriamente desde la poblaci´on dosindividuos(soluciones) yde estosdos se selecciona aquelque al evaluarlo enla funci´on objetivo obtenga el mejor resultadodeseado(maximiza o minimiza).Existe otras variantes no se expondr´an en este documento, sin embargo una muy buena referencia se encuentra en[6] p´ag 67.

Para el caso de la Rueda de la Fortuna los elementos de la nueva generaci´on se obtienenpor mediodelageneraci´onde una variable aleatoriadiscretaque sedeﬁne de la siguiente forma:

1. Se ordena la Poblaci´on Actual de mayor a menor de acuerdo a la valoraci´on de cada cromosoma en relaci´on a la funci´on objetivo.
2. Se suman las evaluaciones de la funci´on objetiva de todos los cromosomas.
3. A cada cromosoma se le asigna unaprobabilidad de ser seleccionada de forma

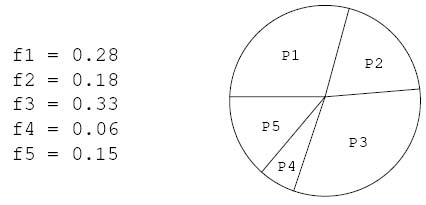
Evaluacion(i)

aleatoria, con Evaluacion = � .

Evaluacion(Poblacion)

Se selecciona un padre y una madre seg´un elpeso dadopor la funci´on de evaluaci´on y la tabla de frecuencias relativas que se construye. Un ejemplo se muestra en la Figura 2.4, que se ha tomado desde S´oley [18], donde se tiene 5 individuos en la poblaci´on, el individuo P1 tiene un valor de ﬁtness de f1, P2 de f2, etc. y donde se observaque sila ruletadiera vuelta sedentendr´ıa con m´asprobabilidad en P3y que con menos frecuencia se obtendr´a P4.

Figura 2.4: Ejemplo del m´etodo de selecci´on Ruleta de la Fortuna.



Criterio de parada

Adem´as de los procesos de cruce y mutaci´on que ya se han explicado, es necesario contar con alg´un criterio de parada del algoritmo. Es claro, que mientras avanza el tiempode ejecuci´ondel AGlas nuevasgeneraciones ser´an cada vez m´ashomog´eneas, por este motivo un criterio natural de parada es detener el procedimiento cuando un gran n´umero de cromosomas de la poblaci´on sean iguales. Otro criterio podr´ıa ser deﬁnir un tiempo de ejecuci´on ﬁjo y luego de la ´ultima iteraci´on elegir aquella

soluci´on representadapor un cromosoma con mejor funci´on de evaluaci´on (ﬁtness). Un seudoc´odigo cl´asico y simple del AG se muestra a continuaci´on:

Procedimiento Algoritmo Gen´etico Generar poblaci´on inicial Evaluar poblaci´on

While NO Criterio de Parada Selecci´on\_poblaci´on Cruzar poblaci´on Mutar poblaci´on Evaluar poblaci´on

end While

2.2. Algoritmo Gen´etico para Calendarizaci´on

2.2.1. Elementos fundamentales en la Calendarizaci´on Estructura del Cromosoma

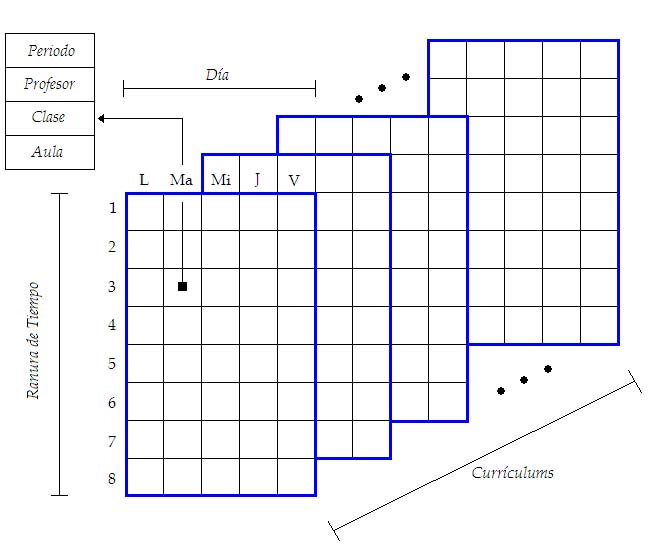
Para el problema de calendarizaci´on, en particular para el CB-CTT cada cromo­soma, representa un calendario completo. Como ya se ha discutido, el cromosoma est´a constituidopor una cadenade genes y en este caso cada gen es una estructura compuesta al menos de cinco componentes:

1. D´ıa
2. Ranura de tiempo
3. Profesor
4. Clase(de algunaMateria)
5. Aula

Recordemosque elpar(D´ıa, Ranura de tiempo)forman un periodo. Un ejemplo se ilustra en la Figura 2.5. Estos 5 componentes se deﬁnieron en la Secci´on 1.3.1.

Muchos autores optan por diferentes representaciones, por ejemplo, en [16] se re­presentaunindividuopormediodeunamatrizden´umerosenteros,dondecada ﬁla representa respectivamente un vector de profesores, periodos y cursos. Estos tres vectores sondeﬁnidosportodaslasposibles combinacionesdeestas entidades.Si se tienen dos profesores, tres periodos y tres materias y si se supone que el profesor 0 puedeimpartirlas materias0y1, elprofesor 1lasmaterias 0y2,dela mismaforma el profesor 0 est´a disponible en los periodos 0y 1, mientrasque el1 est´adisponible en los tres periodos, por otro lado si las materias 0, 1 y 2 pueden ser programados en los tres periodos, la matriz con las tres ﬁlas vectores se ver´ıan como:

Figura 2.5: Representaci´on de un cromosoma para el problema de calendarizaci´on.



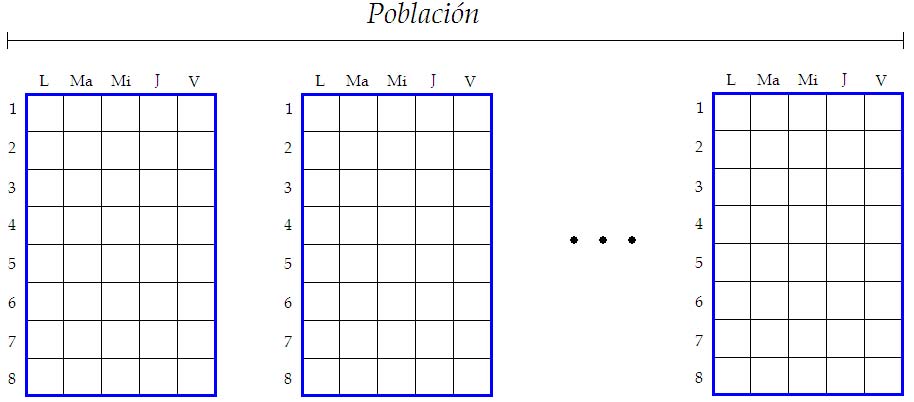
Vector de Profesores: (0,0|0,0|1, 1|1, 1|1, 1) Vector de Periodos: (0,0|1,1|0, 0|1, 1|2, 2) Vector de Materias: (0,1|0,1|0, 2|0, 2|0, 2)

De esta forma el profesor 0puededar clases en elperiodo0y 1,las materias 0y 1, y cada bloque representar´a entonces el horario disponible para cada profesor.

Estructura de la Poblaci´on Inicial

La Poblaci´on Inicial est´a formada por un conjunto de calendarios, generados de forma aleatoria,tratandode obtenerdiversidady calidad enlasposibles soluciones, ver Figura 2.6.

Figura 2.6: Ejemplo de Poblaci´on Inicial para Calendarizaci´on.



Funci´on de evaluaci´on

Para este trabajo se considera una funci´on que cuente las penalizaciones en las que se incurre si se viola alguna restricci´on, tanto suaves como duras, basado en las medidas que se mencionan en la Secci´on 1.3.2. Un resumen se da en la Tabla 2.1. En otras palabras esta funci´on de evaluaci´on (ﬁtness)de un cromosoma determina que tan mala es la soluci´on, y si esta medida es alta, tendr´a menos probabilidad de que su informaci´on gen´etica pase a generaciones futuras.

Tabla 2.1: Resumen de penalizaciones por incumplimiento de restricciones.

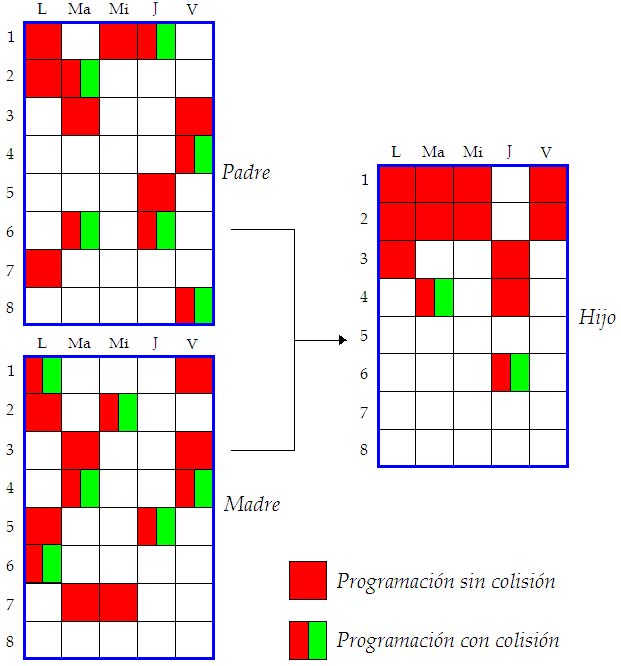
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Restricciones | Caso | | | | | | | | | | Penalizaci´on (puntos) | | | |
|  | Restricciones Duras | | | | | | | | | |  | | | |
| Lecturas | Lectura no programada | | | | | | | | | | 1 | | | |
| Aulas ocupadas | Lectura extra en una misma aula y periodo | | | | | | | | | | 1 | | | |
| Conﬂictos | Lecturas de la misma materia, curr´ıculum o dictada por un mismo profesor en un mismo periodo | | | | | | | | | | 1 | | | |
| Disponiblidad | Lectura en un periodo no disponible por un profesor | | | | | | | | | | 1 | | | |
|  | Restricciones Suaves | | | | | | | | | |  | | | |
| Capacidad de aulas | Un alumno adicional en un aula con capacidad limitada | | | | | | | | | | 1 por cada estudiante adicional | | | |
| D´ıas de trabajo m´ınimo | Lecturas dictadas en un n´umero de d´ıas no permitido | | | | | | | | | | 5 por cada d´ıa por debajo del m´ınimo | | | |
| Curriculum compacto | Lecturas en un mismo d´ıa no adyacentes | | | | | | | | | | 1 | | | |
| Estabilizaci´on de aulas | Diferente aula para lecturas de una misma materia | | | | | | | | | | 1 | | | |

2.2.2. Operadores gen´eticos en la Calendarizaci´on Operador de Cruce

El cruce entre calendarios padres debe tratar de que sus descendientes posean la informaci´on gen´etica m´as fuerte y que su funci´on de evaluaci´on, tal como se deﬁ­ni´o en 2.2.1, disminuya lo m´as posible. Para esto, este procedimiento debe evitar, en la medida de lo posible, que en sus hijos exista colisiones en la programaci´on, es decir, evitar ubicar materias de un mismo curr´ıculum en un mismo periodo por ejemplo, ver Figura 2.7.

Unadelos resultadosque seha encontrado en estainvestigaci´on, esquelaideadela implementaci´oncomputacionaljueganunrol muyimportanteenesteaspecto. Un algoritmo podr´ıa construir una poblaci´on inicial con cromosomas donde dos aulas diferentes nunca se choquen en un mismoperiodo,yquiz´as otraideadeprogramaci´on procure individuos donde los profesores no se les asignan lecturas en periodos fuera de sudisponibilidad.Inclusopuedehaber unageneraci´onde cromosomas que eviten estos dos problemas mencionados, tal como se pudo lograr en este trabajo y como se ver´a en el siguiente cap´ıtulo.

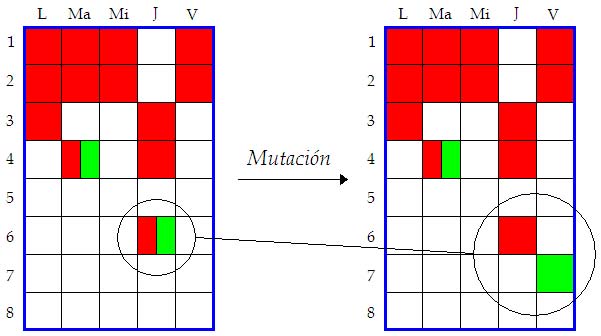
Figura 2.7: Ilustraci´on del operador de cruce que evita colisiones.



Operador de Mutaci´on

El operador de mutaci´on simplemente sigue la idea de cambiar un gen por otro mejor, como se ilustra en la Figura 2.8. El gen que mutar´a es seleccionado de forma aleatoria.Otra idea es incrementar el n´umero de genes a cambiar por medio de generar un n´umero aleatorio discreto menor al largo del cromosoma.

Figura 2.8: Ilustraci´on del operador de mutaci´on.



Cap´ıtulo 3

Implementaci´on computacional de un AG para el CB-CTT

®

3.1. El entorno Matlab R

®

3.1.1. Caracter´ısticas de MatlabR

EnRodr´ıguez[14] sedeﬁne aMatlab®R1 como un instrumento computacional sim­ple, vers´atily degranpoderpara aplicaciones num´ericas, simb´olicasygr´aﬁcas,que contiene una gran cantidad de funciones predeﬁnidas para aplicaciones en ciencias e ingenier´ıa. Esta descripci´on de Matlab muestra muy bien las razones por la cual se ha elegido este lenguaje de programaci´on para la implementaci´on computacional de un AG dise˜nado espec´ıﬁcamente para el CB-CTT.

Las funciones incorporadas en MatlabRon,

® facilitan en gran manera la programaci´porque evita un gasto innecesario al momento de realizar operaciones b´asicas, o no tan b´asicas, que no forman parte del objetivo ﬁnal de un algoritmo espec´ıﬁco. Es­tas funciones van desde el manejo de matrices, hasta aplicaciones en Estad´ıstica e Investigaci´on de Operaciones. Unas de las ventajas en la implementaci´on del AG es el manejo de conjuntos, y el f´acil manejo de estructuras de datos para b´usqueda dentro de vectores de m´as de una dimensi´on, tal como la estructura del cromosma que se propone, ver Figura 2.5.

Alo largo de la literatura sepuede observar el usode estaherramienta endiferentes ´on pa-

areas de la ciencia, para el caso particular de los problemas de calendarizaci´ra centros educativos una referencia se encuentra en Molina [5], donde se utiliza MatlabRondehorariosparala UniversidaddeValparaısoenChile,

® paralageneraci´´

®

aunque la interfaz gr´aﬁca ﬁnal se la realiza en utilizando el lenguaje Visual BasicRpara obtener una salida de calendarios m´as soﬁsticada.

R

Algunas propiedades que caracterizan a Matlab ® tambi´en se mencionan en[14]:

C´alculo num´erico r´apido y con alta precisi´on.

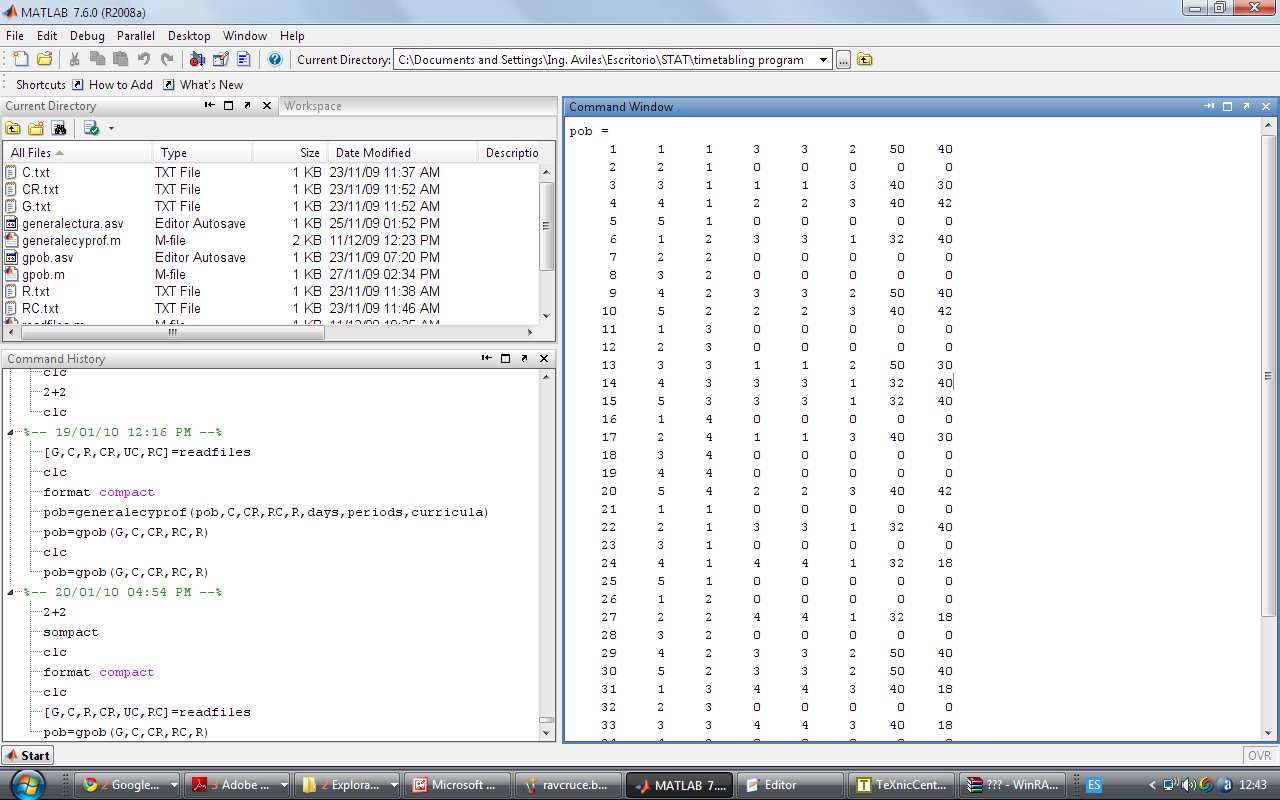
R

1Matlab® es una marca registrada de The MathWorksTM .

Capacidad para manejo matem´atico simb´olico.

Funciones para graﬁcaci´on y visualizaci´on avanzada. Programaci´on mediante un lenguaje de alto nivel. Soporte para programaci´on estructurada y orientada a objetos. Facilidades b´asicas para dise˜no de interfaz gr´aﬁca. Extensa biblioteca de funciones. Paquetes especializados para algunas ramas de ciencia e ingenier´ıa. Sistema de ayuda en linea. Iteraci´on con otros entornos.

Figura 3.1: Entorno MatlabR.

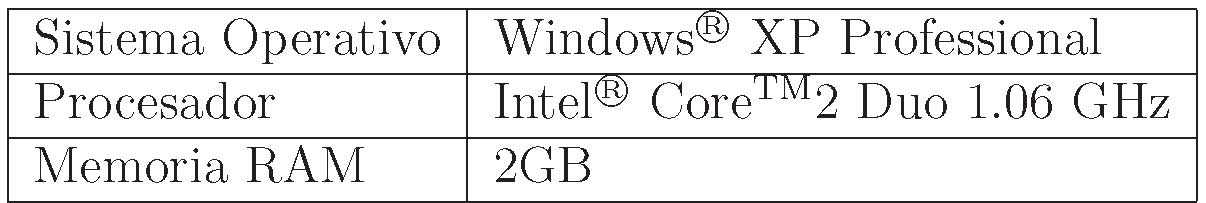
®

3.1.2. Recursos computacionales empleados

La versi´on de MatlabR

® que se ha utilizado es Matlab 7.6.0.324(R2008a), y los recursos de hardware y software se detallan en la Tabla 3.1.

Tabla 3.1: Recursos para implementaci´on computacional del AG



3.2. Datos base para implementaci´on

3.2.1. Formato de datos de entrada

La base de datos es de mucha importancia para esta implementaci´on, porque el c´odigo llama a la informaci´on inicial desde archivos de formato plano txt para em­pezar elprocesode calendarizaci´on,que se relacionan directamente con un formato matricial deﬁnido espec´ıﬁcamente para este prop´osito.

Unosdatosde entradaque sirvendepruebapara analizarla eﬁcienciade heur´ısticas y metaheur´ısticas para problemas de Calendarizaci´on(scheduling) y en particular para problemas de tipo CB-CTT sepueden encontrar enlap´agina webdel SaTT: Scheduling andTimetablingGroup2, aqu´ısedeﬁne unformatodedatosparainicio al que llaman ECTT (Extended Curriculum-based Course TimeTabling for­mat),justamentepara el CB-CTT,que contiene tablas con informaci´on espec´ıﬁca que guardan relaci´on con el formato requerido para la competici´on mundialmente conocida ITC (Internacional Timetabling Competition)De Cesco[1], donde los participantes deben remitirse a esta forma de ingreso, porque entre otras cosas, permite hacer ejecuciones de prueba para algunos ejemplos generados para este ﬁn y compartir experiencia entre investigadores de forma estandarizada.

Las tablas a las que hacemos referencia, contienen las siguientes variables de inicio:

Tabla: Datos generales

Nombre para el problema a ejecutar.

N´umero de aulas.

N´umero de d´ıas para calendizar.

N´umero de periodos por d´ıas.

N´umero de curr´ıculums.

Min-Max lecturas diarias.

N´umero de restricciones de no disponibilidad.

N´umero de restricciones de aulas.

Tabla: Materias

Nombre de identiﬁcaci´on para las materias.

Nombre de los profesores que dictan las materias.

N´umero de lecturas de cada materia.

2Diegm-UniversityofUdine(Italy), main contacts:Prof.AndreaSchaerfyDr.LucaDiGaspero; http://tabu.diegm.uniud.it/

M´ınimo n´umero de d´ıas para programar a las materias.

N´umero de estudiantes que toman una materia.

Valor binario 0-1 que indica si se requiere o no que dos lecturas programadas en el mismod´ıadeban ser asignadas aperiodos consecutivo(algoque ocurre en la mayor´ıa de los casos).

Tabla: Aulas

Nombre de identiﬁcaci´on de las aulas. Capacidad de las aulas. Identiﬁcadorquedeﬁnela ubicaci´on(en unbloquede ediﬁciopor ejemplo) de

cada aula.

Tabla: Curr´ıculums

Nombredeidentiﬁcaci´ondel curr´ıculum(semestre). N´umero de cursos de los curr´ıculums. N´ombres de identiﬁcaci´on de las materias en cada curr´ıculum.

Tabla: Restricciones de no disponibilidad

Nombre de identiﬁcaci´on para las materias. D´ıas en que las materias no pueden ser programadas. Periodosdeld´ıa(Timeslots) enquelas materias nopueden serprogramadas.

Tabla: Restricciones de aulas

Nombre de identiﬁcaci´on para las materias.

Nombre de identiﬁcaci´on del aula donde no pueden ser programadas las ma­terias.

3.2.2. Formulaciones

Adem´as de las tablas mencionadas en la secci´on anterior, en el ITC se caracterizan losproblemasaresolverpor mediodecincodiferentesformulaciones(verTabla3.2) que contiene un espec´ıﬁco conjunto de componentes de costos y los pesos (penali­zaciones de acuerdo a la Tabla 2.1) correspondiente a cada uno de ellos, en realci´on a la deﬁnci´on de la funci´on objetivo que tan solo suma los costos por violaci´on de restricciones.

En la Tabla 3.2, UD se reﬁere a la Universidad de Udine de Italia, la restricci´on Curr´ıculum compacto Versi´on dos se deﬁne como: Lecturasprogramadas en un mis­mo d´ıa no deben tener lecturas de otras materias entre ellas y “D” es la inicial de restricciones duras.

Tabla3.2:Descripci´ondeformulacionesdeproblemasde calendarizaci´on. Adaptado deDeCesco[1].Para referenciadelos nombresdeloscomponentesde costo considere la Secci´on 1.3.2

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Formulaci´on del problema: Componente de costos | | | | | UD1 | | UD2 | | UD3 | UD4 | | UD5 |
| Lecturas | | | | | D | | D | | D | D | | D |
| Conﬂictos | | | | | D | | D | | D | D | | D |
| Aulas ocupadas | | | | | D | | D | | D | D | | D |
| Disponibilidad | | | | | D | | D | | D | D | | D |
| Capacidad de aulas | | | | | 1 | | 1 | | 1 | 1 | | 1 |
| D´ıas de trabajo m´ınimo | | | | | 5 | | 5 | | - | 1 | | 5 |
| Curr´ıculum compacto | | | | | 1 | | 2 | | - | - | | 1 |
| Curr´ıculum compacto versi´on dos | | | | | - | | - | | 4 | 1 | | 2 |
| Estabilizaci´on de Aulas | | | | | - | | 1 | | - | - | | - |
| Carga m´ınima y m´axima | | | | | - | | - | | 2 | 1 | | 2 |
| Distancia de traslado | | | | | - | | - | | - | - | | 2 |
| Idoneidad de aulas | | | | | - | | - | | 3 | D | | - |
| Lecturas dobles | | | | | - | | - | | - | 1 | | - |

La formulaci´on UD4 es de importancia para este trabajo, porque considera todas las restricciones que se espera se cumplan en la aplicaci´on del AG que se propone. Adicionalmente se cumple tambi´en la restricci´on Curr´ıculum Compacto versi´on 1.

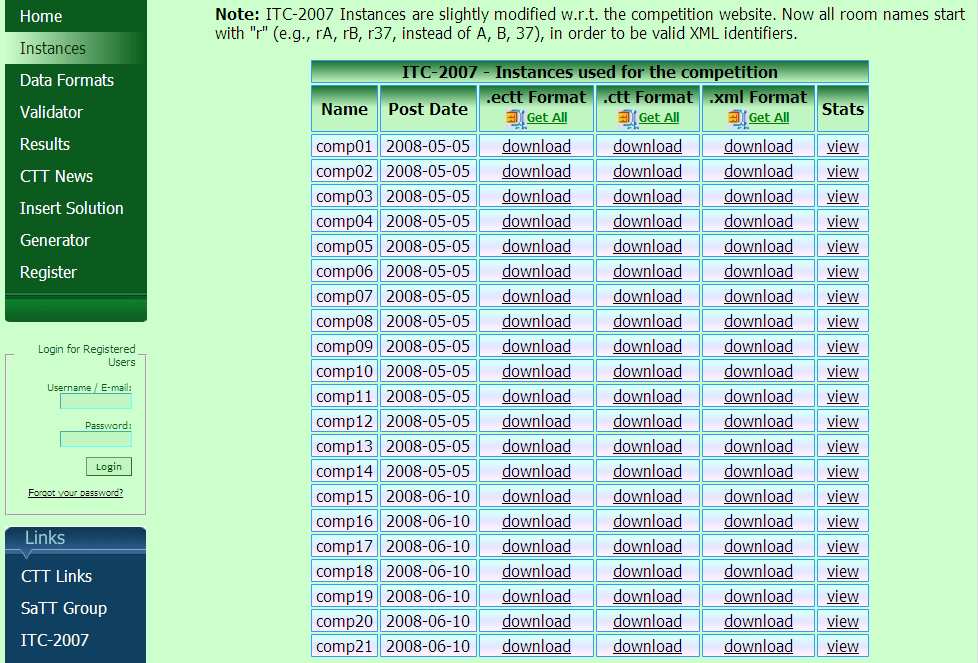
3.2.3. Instancias

En la p´agina web del SaTT se puede encontrar una serie de problemas bajo las formulaciones dadas en la Secci´on 3.2.2 que son instancias que pueden bajarse de forma gratuita para hacer pruebas de alg´un algoritmo propuesto para rersolver el problema de Calendarizaci´on Educacional, 21 de estas instancias corresponden a datos reales tomados de la Universidad de Udine, codiﬁcadas con los nombres comp01, ··· ,comp21, ver Figura 3.2.

Enlasinstancias consta adem´as unpeque˜no ejemplodenominado Toy que es unode los pocos problemas donde se encuentra un valor de la funci´on de evaluaci´on igual a cero. Este valor ´optimo se lo ha realizado bajo la t´ecnica del Recocido simulado que se menciona en la Secci´on 1.2.2.

Toy se ha utilizado para hacer las pruebas de ejecuci´on de nuestro AG y sus com­ponentes se describe en un archivo txt que sedetalla enlaTabla3.3yqueguardan relaci´on a lo deﬁnido en el subcap´ıtulo 3.2. Toy puede encontrase en cualquiera de las formulaciones descritas en la Secci´on 3.2.2, sin embargo para las corridas del programa se ha tomado la versi´on ECTT.

Figura 3.2: Instancias de la Universidad de Udine vista desde la p´agina Web de SaTT.



®

3.2.4. Matrices de entrada para el AG en MatlabR

Conelprop´ositodedarcaracter´ısticas ﬁjasalosdatosdeentrada,sehadise˜nadoun esquema(template)paraque elalgoritmoque seproponepuedaejecutarlasfuncio­nesyprocedimientos ´odigoMatlabR

escritas en c´®. Este esquema sigue el formato de datos de entrada descrito en el apartado 3.2.1, con el ligero cambio en los nombres delos profesores, c´odigode materias yaulas, alasque seha representado utilizando n´umeros dentro del conjunto N = {1,2, 3,..., ∞}.

Se han creado archivos de extensi´on txt que ser´an le´ıdos por un simple procedi­miento cuyo c´odigo sepuede revisar en elPrograma1.Los nombresy un c´odigode identiﬁcaci´on de estos archivos planos se detallan en la Tabla 3.4.

Programa 1 “readﬁles.m”: Procedimiento para leer archivos externos txt

G=load(’G.txt’); C=load(’C.txt’); R=load(’R.txt’); CR=load(’CR.txt’); UC=load(’UC.txt’); RC=load(’RC.txt’);

Tabla 3.3: Contenido del archivo toy.txt

Name: Toy Courses: 4 Rooms: 3 Days: 5 Periods\_per\_day: 4 Curricula: 2 Constraints: 8

COURSES: SceCosC Ocra 3 3 30 ArcTec Indaco 3 2 42 TecCos Rosa 5 4 40 Geotec Scarlatti 5 4 18

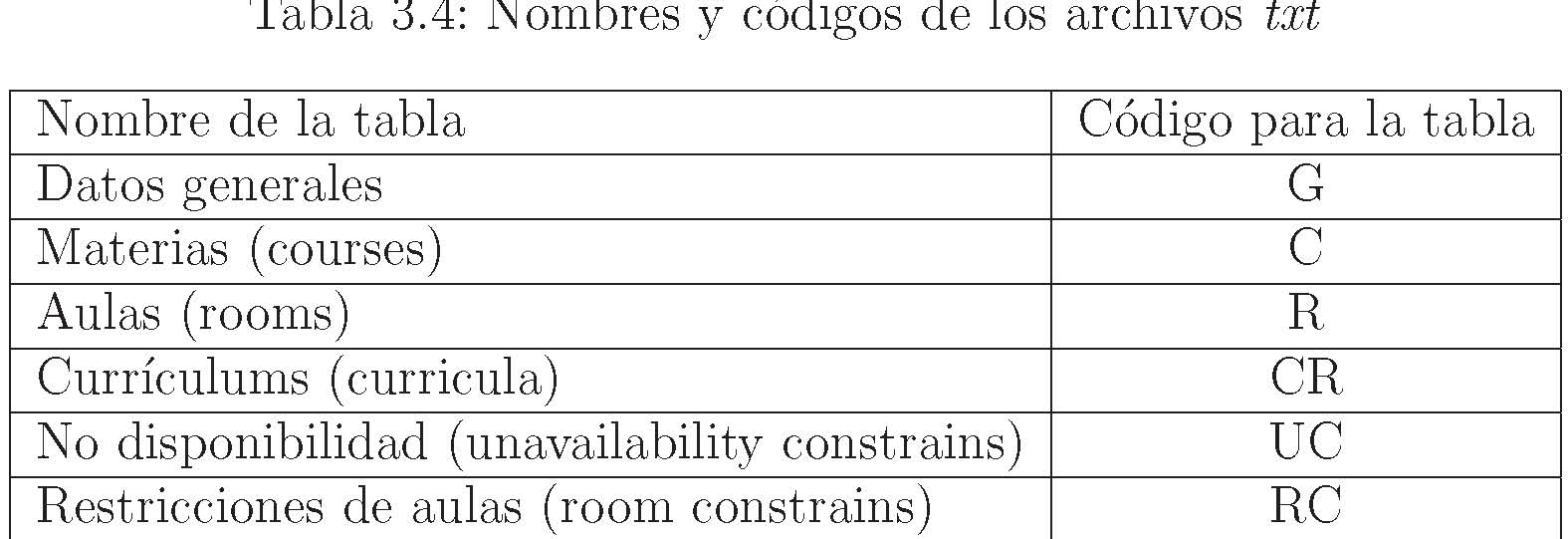
ROOMS: rA 32 rB 50 rC 40

CURRICULA: Cur1 3 SceCosC ArcTec TecCos Cur2 2 TecCos Geotec

UNAVAILABILITY\_CONSTRAINTS: TecCos 2 0 TecCos 2 1 TecCos 3 2 TecCos 3 3 ArcTec 4 0 ArcTec 4 1 ArcTec 4 2 ArcTec 4 3

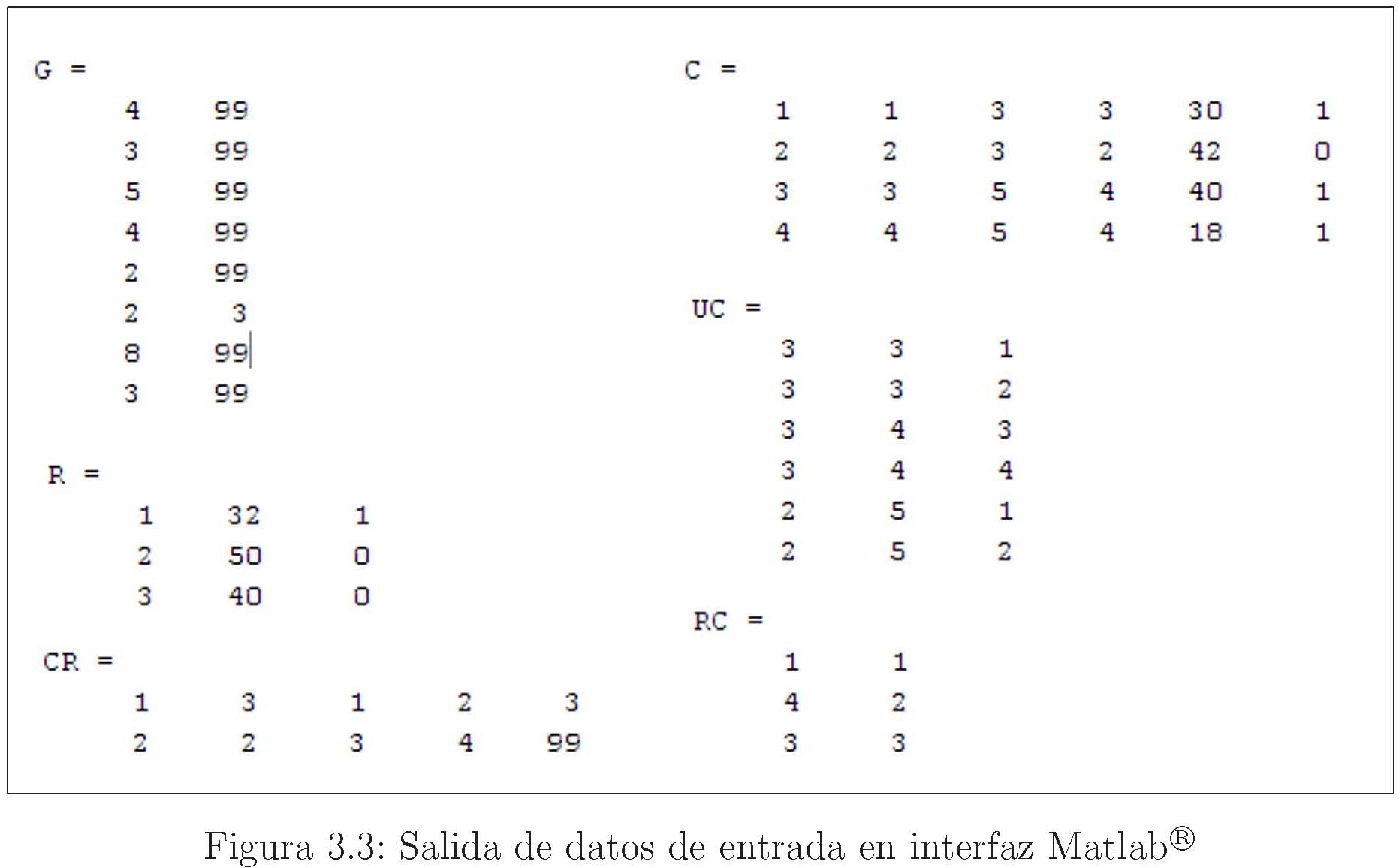
ROOM\_CONSTRAINTS: SceCosC rA Geotec rB TecCos rC

END.



Las matrices generadas por el Programa 1, se muestran en pantalla como matrices de n´umeros, los cuales son f´aciles de manipular para ejecutar y apropiadas para otras funciones del AG, ver Figura 3.3.

Figura 3.3: Salida de datos de entrada en interfaz MatlabR

®

Los n´umeros99 representan espacios vaciosdentro enlas matricesformadasporlos datos de los archivos de entrada txt que son llamados por el Programa 1.

®

3.3. Dise˜no del AG para el CB-CTT en Matlab R

3.3.1. Representaci´on cromos´omica del calendario

Un calendario se ha representado en este trabajo como una matriz de dimensi´on [(d´ıas)(ranuras de tiempo por d´ıa)(curr´ıculums) × 8]. En este arreglo, el n´umero de ﬁlas corresponde a cada periodo disponiblepara calendarizar multiplicadopor el n´umero de curr´ıculums considerados en un problema particular, esto quiere decir que los calendarios de cada curr´ıculum se dispondr´an uno debajo del otro. Por otra parte, cada columna de la matriz representa respectivamente:

1. Indicador del n´umero de la ﬁla 1, 2, 3,..., etc.
2. D´ıadela semana(laborables).
3. Ranuradetiempo(horasde clasesdisponibles).
4. Materia que sedictar´a en un periodo.
5. Profesor de la materia.
6. Aula a utilizar.
7. Capacidad del aula a utilizar.
8. N´umero de estudiantes registrados en la materia.

´

IndiceD´ıaRanuraMateriaProfesorAulaCapacidad

Estudiantes

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | | | 1 | 1 | 3 | 3 | 2 | 50 40 | | |
| 2 | | | 2 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 3 | | | 3 | 1 | 1 | 1 | 3 | 40 30 | | |
| . | | | . | . | . | . | . | . | . | |
| . | | | . | . | . | . | . | . | . | |
| . | | | . | . | . | . | . | . | . | |
| 19 | | | 4 | 4 | 0 | 0 | 0 | 0 | 2 | |
| 20 | | | 5 | 4 | 2 | 2 | 3 | 40 42 | | |
| 21 | | | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 22 | | | 2 | 1 | 3 | 3 | 1 | 32 40 | | |
| . | | | . | . | . | . | . | . | . | |
| . | | | . | . | . | . | . | . | . | |
| . | | | . | . | . | . | . | . | . | |
| 38 | | | 3 | 4 | 4 | 4 | 3 | 40 18 | | |
| 39 | | | 4 | 4 | 0 | 0 | 0 | 0 | 0 | |
| 40 | | | 5 | 4 | 3 | 3 | 1 | 32 40 | | |

Figura3.4:Representaci´on matricialde un cromosoma(calendario)

Un ejemplo de la representaci´on matricial de un cromosoma para Toy se observa en la Figura 3.4, donde los diferentes colores rojo y azul determinan respectivamente un calendario para el curr´ıculum 1y curr´ıculum 2.

Una de las ventajas de esta representaci´on matricial es que no permite que lecturas de una misma materia sean programadas en un mismo periodo, es decir, la restric­ci´on dura alaque sehallamado Lecturas, verSecci´on1.3.2, se superapor completo, ytodaposiblesoluci´on alproblemaplanteadotendr´a cero como valordeevaluaci´on

para esta restricci´on.

De esta mismaforma ninguna materiade un mismocurr´ıculum se ubica en un mismo periodo, esdecir, se aseguraque Conﬂictos,la cual es otra restricci´on dura, se cumpla en un50%,porques´ıpodr´ıaproramarsedos materiasdediferente curr´ıculum en una misma combinaci´on (d´ıa,ranura de tiempo) pero con diferente profesor y en diferente aula.

3.3.2. Algoritmo para crear un cromosoma

Lafunci´on gcrom,verPrograma9,Ap´endice A,sehacreadoparagenerarlarepre­sentaci´on matricialde un cromosoma ubicando aleatoriamentepara cada curr´ıculum en alg´un periodo una materia, un profesor y un aula, adem´asde su capacidady del n´umero de estudiantes registrados, tom´andolas desde los archivos txt’spreviamente llamados por la funci´on readﬁles, ver Programa 1. La funci´on tiene cuidado de no ubicar m´asdelas materiasdeﬁnidaspara cada curr´ıculum en el archivo CR.txt, ver Tabla 3.4. Todos esto se logra con una funci´on denominada generalecyprof que a su ves llama a un procedimiento llamado roomycap, ver Programas 12 y 10 en el Ap´endice A.

En estepunto se aplica unprocedimientoque tratade mejorar el cromosoma inicial cambiando aleatoriamente el aula antes asignadapor otra siemprey cuandola fun-ci´on objetivo mejore en su valor, para este caso esto quiere decir que la restricci´on suave llamada Capacidad de Aulas se supere. Este procedimiento se ha nombrado como mejoraroom, ver Programa 11 tambi´en en el Ap´endice A.

3.3.3. Programa de generaci´on de la Poblaci´on Inicial

Laprimerageneraci´on se construye simplemente repitiendo n veceslafunci´on gcrom, donde n es el tama˜no de la poblaci´on. Como ya se ha mencionado, se pretende que esta poblaci´on contenga soluciones diversas pero tambi´en de calidad.

La funci´on gpob se muestra en el Programa 2.

Programa 2 “gpob.m”: Funci´on que genera la poblaci´on inicial

function poblacion=gpob(G,C,CR,RC,R,UC,n) for i=1:n poblacion(:,:,i)=gcrom(G,C,CR,RC,R,UC); end

3.3.4. Forma de la funci´on de evaluaci´on

Todas las restricciones descritas en la Secci´on 1.3.2 se han medido considerando las penalizaciones resumidas en la Tabla 2.1. La funci´on ﬁtness, que se presenta en el Programa 3, llama a los procedimientos roomoccupancy, conﬂicts, ucons­traints, roomcapacity, minways, ccompactness y rstability, que representan a las restricciones que se consideran para el problema del CB-CTT, ver Tabla 3.5.

Programa 3 “ﬁtness.m”: Funci´on de evaluaci´on para un cromosoma

function [ffitnessv ffitness]=fitness(cromosoma,G,R,UC,C)

days=G(3,1);

periods=G(4,1);

rooms=G(2,1);

courses=G(1,1);

curricula=G(5,1);

le=days\*periods\*curricula;

roomoccupancy;

conflicts;

uconstraints;

roomcapacity;

minwdays;

ccompactness;

rstability;

ffitnessv=[froomoccupancy fconflicts fuconstraints froomcapacity fminwdays fccompactness frstability];

ffitness=sum(ffitnessv);

Tabla 3.5: Procedimientos de la funci´on ﬁtness

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de procedimientos | Nombre de restricciones | | |
| roomoccupancy | Aulas Ocupadas | | |
| conﬂicts | Conﬂictos | | |
| uconstraints | Unavailability constraints-Disponibilidad | | |
| roomcapacity | Capacidad de aulas | | |
| minwdays | D´ıas de trabajo m´ınimo | | |
| ccompactness | Curr´ıculum Compacto | | |
| ccompactnessv2 | Curr´ıculum Compacto versi´on dos | | |
| rstability | Estabilizaci´on de aulas | | |

Un ejemplo de la aplicaci´on de la funci´on en MatlabR

on de evaluaci´® se presenta en la Figura 3.5. La suma del vector ﬁtnessv es la medida de incumplimiento de restricciones.

3.3.5. Evaluaci´on de una poblacion

La evaluaci´on en la poblaci´on se la realiza utilizando la funci´on evaluapoblacion, ver Programa 4. Este procedimiento simplemente da el valor de ﬁtness para cada

>> [ffitnessv ffitness]=fitness(cromosoma,G,R,UC,C) ffitnessv = 1 0354 22 04 ffitness = 66

Figura 3.5: Ejemplo de aplicaci´on de la funci´on ﬁtness

cromosoma,present´andolos en un vectorque sehallamado ﬁtnessp, verFigura3.6.

Programa 4 “evaluapoblacion.m”: Devuelve un vector con el ﬁtness de cada cromosoma de una poblaci´on

function [fitnessp]=evaluapoblacion(poblacion,G,R,UC,C) s=size(poblacion); tp=s(3); for i=1:tp

fitnessp(i,1)=fitness(poblacion(:,:,i),G,R,UC,C); end

>> [fitnessp]=evaluapoblacion(poblacion,G,R,UC,C)

fitnessp =

36

58

76

71

87

Figura 3.6: Ejemplo del vector ﬁtnessp para una poblaci´on de tama˜no 5

3.3.6. El proceso de selecci´on

Para poder pasar de generaci´on en generaci´on se ha dise˜nado un procedimiento de selecci´on de soluciones que permite elegir de entre todos los cromosomas de una poblaci´on alosmejores, aquelloscuyafunci´ondeevaluaci´onsealamejorposible.El m´etodoque sehaimplementado esla Rueda de la Fortuna (Roulette Wheel)que se deﬁni´o en el Apartado 2.1.3, ver Programa 5.

3.3.7. Dise˜no del procedimiento de cruce

En la funci´on cruce, ver Programa 20, Ap´endice A, se desarrolla un procedimiento sistem´atico, que transﬁere informaci´on gen´etica de un padre y una madre para ge­nerar un hijo y una hija.

Elproceso comienzabuscando en elcalendario delpadre ylamadre el periodo donde cada una de las materias ya hayan sido prgoramadas. Luego se genera un n´umero aleatoriouniformedepar´ametros1y0 conel ﬁndedeterminarsi esel gen del padre que se tranferir´a al hijo o el de la madre, y se ubicar´an los datos del ascendiente en el mismo periodo encontrado, cuidando que esta ubicaci´on est´e libre en el descen­diente, caso contrario se selecciona una ubicaci´on libre de forma aleatoria. Cuando la elecci´on se ha realizado, la informaci´on que se tranﬁere al otro descendiente es exactamente lo opuesto, siempre que sea posible. Todo esto se lo realiza para cada calendario de cada curr´ıculum.

Un ejemplo del proceso de cruce se muestra en la Figura 3.7. El ejemplo ilustra el paso de informaci´on gen´etica hacia los hijos utilizando los colores azul y rojo para el padre y la madre respectivamente, ubicando los genes en las ubicaciones corres­pondientes de acuerdo a lo mencionado en el p´arrafo anterior.

3.3.8. Funci´on de cruce para la Poblaci´on

Con el prop´osito de asegurar que las futuras generaciones posean miembros m´as fuertes, se ha decidido que al momento de cruzar dos individuos, se elija entre el

Programa 5 “seleccion.m”: Selecciona individuos para nueva generaci´on

function poblacion=seleccion(poblacion,fitnessp) sp=size(poblacion); maximo=max(fitnessp); veclargo=[]; for i=1:sp(3);

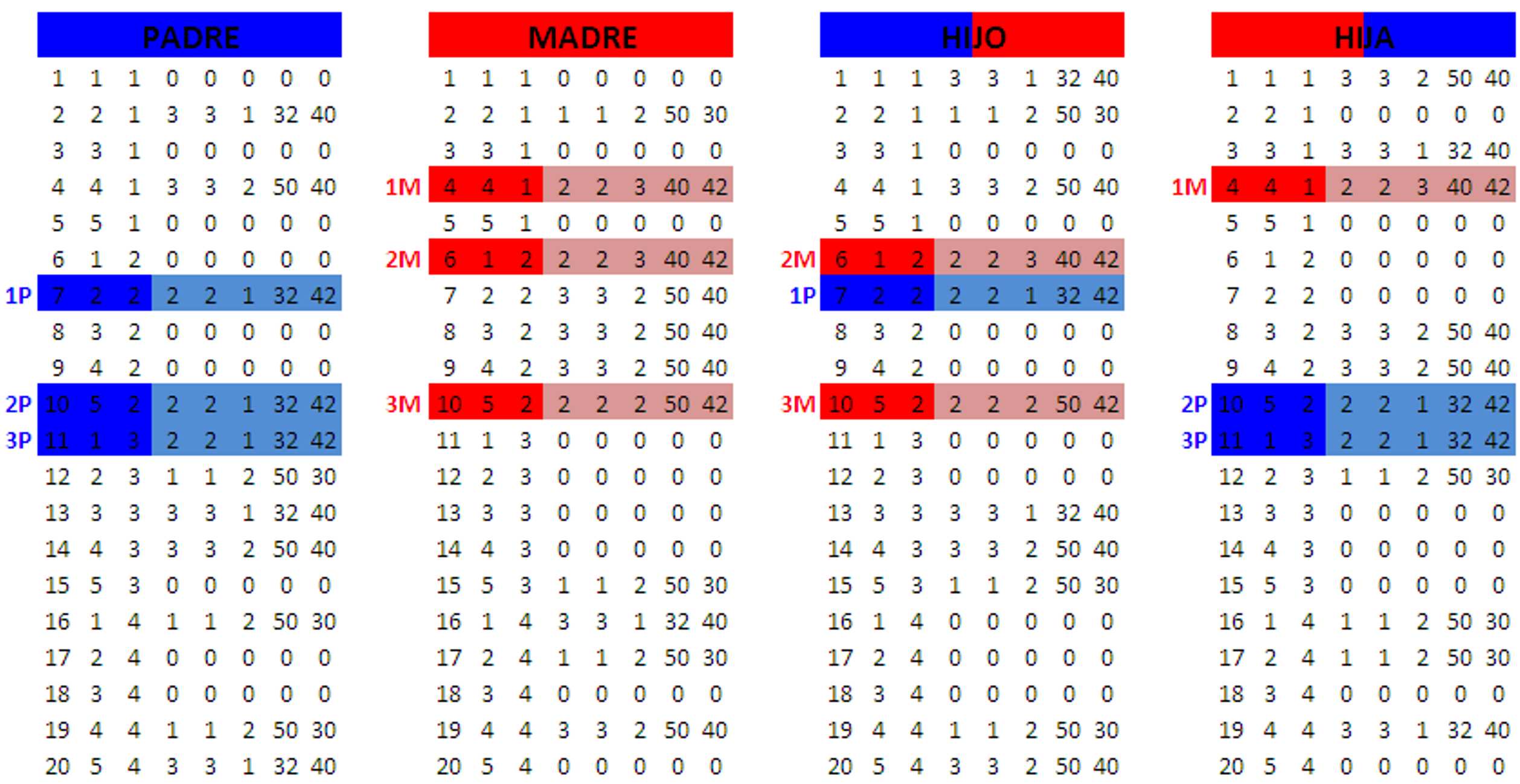
vecs(i)=maximo-fitnessp(i,1)+1; for k=length(veclargo)+1:length(veclargo)+vecs(i) veclargo(k)=i;

end end for k=1:sp(3)

r=ceil(length(veclargo)\*rand); v=veclargo(r); nuevapoblacion(:,:,k)=poblacion(:,:,v);

end poblacion=nuevapoblacion;

Figura 3.7: Ejemplo de un proceso de cruce para la base de datos Toy.



hijo y la hija, al de menor funci´on de evaluaci´on para que forme parte de la nueva poblaci´on. La funci´on ﬁtness se aplica a ambos cromosomas y se veriﬁca quien so­brevive y quien no. El padre y la madre se seleccionan de forma aleatoria desde la poblaci´on luego del proceso de selecci´on descrito en la Secci´on 3.3.6.

Lafunci´onquepermite esteprocedimiento es crucepoblaci´on, verPrograma23 en elAp´endice A.Elpar´ametro probcrucerepresentalaprobabilidaddequedosindi­viduos se crucen,y se ajustaluegode experimentos computacionales como se ver´a en el siguiente cap´ıtulo. En la literatura que se ha revisado este valor regularmente es alto.

3.3.9. Mutaci´on de un cromosoma

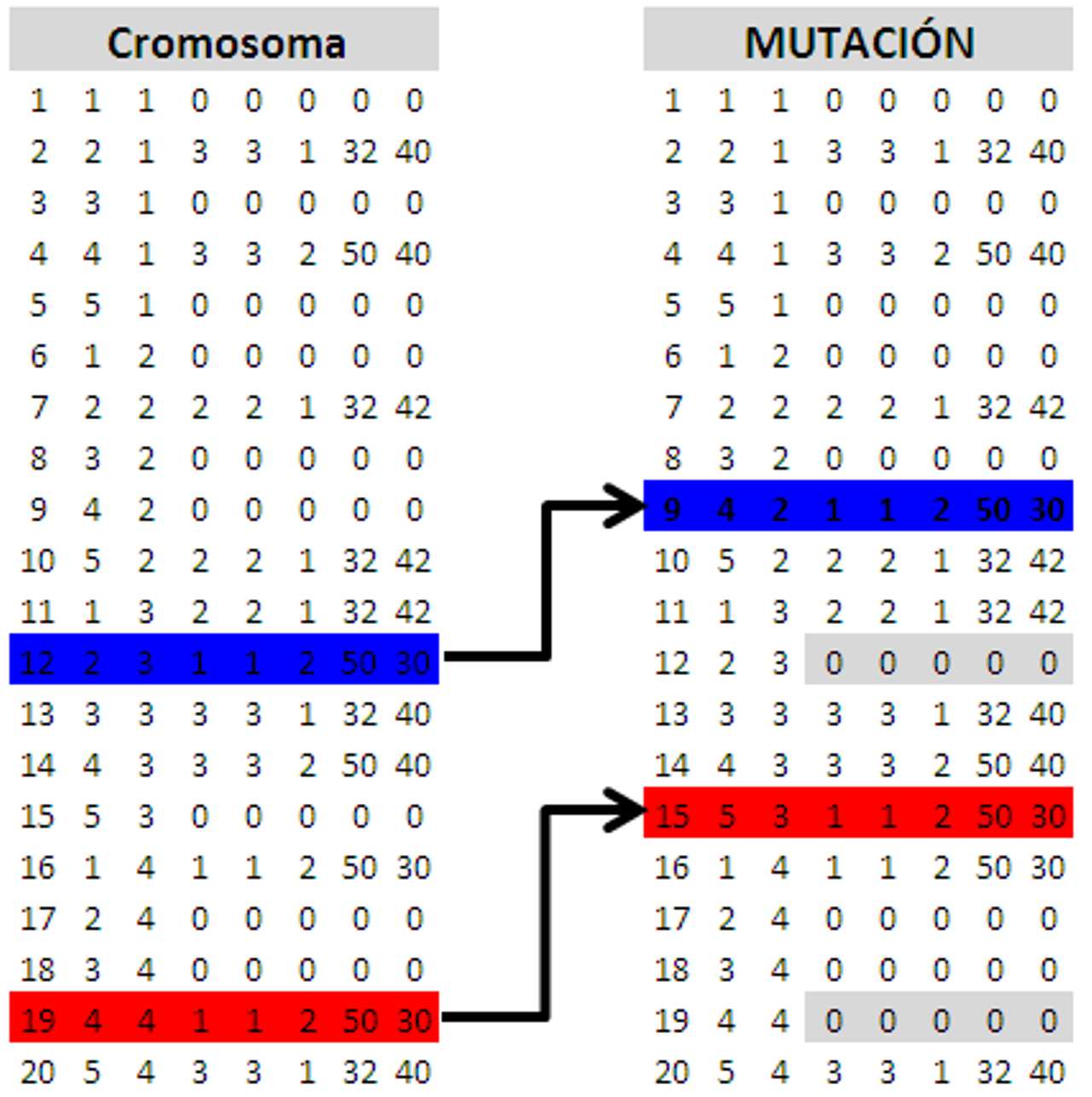
La mutaci´on es unprocedimientoque act´ua sobrelos genes.En estaimplementaci´on se ha considerado para cada ﬁla de la representaci´on cromos´omica no vac´ıa, ubicar el gen (la ﬁla) en un lugar disponible de forma aleatoria, pero en este caso, este cambio se produce siempre que la funci´on de evaluaci´on mejore en su medida. Esto asegura que si la mutaci´on en un gen ocurre, esto sea para bien, ver Figura 3.8.

´

El valordelafunci´onde evaluaci´onpara Cromosoma es 97ypara MUTACI ON es 93.

La funci´on mutacion produce estos cambios, ver Programa 24 en Ap´endice A. En el c´odigo sepuede observar,que unpar´ametro de entrada es probmutacion el cual sirve para determinar si un gen es candidato a mutar.

Figura 3.8: Ejemplo de un proceso de mutacion para la base de datos Toy.



3.3.10. Funci´on de mutaci´on para la poblaci´on

Elprocesode mutaci´on debe aplicarse a cada elemento dela poblaci´on de acuerdo a unaprobabilidadque sedeﬁne enlafunci´on mutacionpoblacion, verPrograma25 Ap´endice A, como probmutacion, esta es la misma probabilidad dada en la funci´on mutacion para decidir si un gen de un cromosoma mutar´a o no.Esto signiﬁcaque la misma probabilidad servir´a para deﬁnir si un cromosoma entra al proceso de mutaci´on adem´as de decidir si un gen cambiar´a de posici´on. Este par´ametro debe ajustarse tambi´enluegodeprobar condiferentes valores en muchas corridasdel AG.

3.3.11. Criterio de parada del AG

En teor´ıa el AG podr´ıa ejecutarse en tiempo indeﬁnido, teniendo en consideraci´on que los elementos de la poblacion luego de un tiempo t grande se parecer´an entre s´ıy se espera adem´asquelafunci´onde evaluaci´ontenga un valor cercano a cero(o cero en el mejor de los casos). Debido a esto un criterio de parada del AG se ha deﬁnido en este trabajo como:

1. Para si al menos una proporci´on p de individuos miembros de la poblaci´on tienen el mismo valor de ﬁtness, ´o
2. Si enalgunaiteraci´onalg´unelementodela poblaci´on tiene un valordefunci´on de evaluaci´on igual cero

En el primer caso la soluci´on ser´a aquel cromosoma con m´ınimo ﬁtness, si existe m´as de una soluci´on encontrada, se elige una de forma aleatoria, y en el segundo caso aquel que tenga el valor de cero. Enlafunci´on criterioparada, verPrograma6, seimplementa el caso1y en el caso dos se lo considera en el programa general del AG, ver Programa 7.

Programa 6 “criterioparada.m”: Criterio de parada caso 1, proporci´on de ele­mentos con igual ﬁtness

function cparada=criterioparada(fitnessp) if sum(ismember(fitnessp,min(fitnessp)))>=0.8\*length(fitnessp)==1 cparada=1; else cparada=0; end

3.3.12. AG completo

Luegode tener todoslos elementos necesariosparalaimplentaci´ondel AGsolofalta enlazar todos estas rutinas en un solo programa, que permita obtener la soluci´on deseada. A este programa general se denomina agcbctt, ver Programa 7. El cual guarda relaci´on estricta con el c´odigo cl´asico de un AG dado en la Secci´on 2.1.3.

Programa 7 “agcbctt.m”: AG completo

function [poblacion tt]=agcbctt(G,C,R,CR,UC,RC,n,probcruce,probmutacion) poblacion=gpob(G,C,CR,RC,R,UC,n); fitnessp=evaluapoblacion(poblacion,G,R,UC,C); iteracion=0; while (criterioparada(fitnessp) || min(fitnessp)==0) ~=1

poblacion=seleccion(poblacion,fitnessp); poblacion=crucepoblacion(poblacion,G,R,UC,C,probcruce); poblacion=mutacionpoblacion(poblacion,G,R,UC,C,probmutacion); fitnessp=evaluapoblacion(poblacion,G,R,UC,C); iteracion=iteracion+1

end posmin=find(fitnessp==min(fitnessp)); tt=poblacion(:,:,posmin);

Cap´ıtulo 4

Experimentos computacionales

4.1. Calibraci´on de par´ametros

Los valores de los par´ametros de entrada del AG deben ser ajustados mediante un proceso de ensayo y error, es decir, probar diferentes valores hasta que el dise˜nador considerequeha encontrado estabilidad enla ejecuci´ondel algoritmo.Con estabili­dad se quiere decir que el programa ejecutado alcanza los objetivos deseados en un tiempo razonablemente peque˜no. Un resumen de los par´ametros de entrada se detallan a continuaci´on:

1. Tama˜no de la poblaci´on.
2. Probabilidad de mutaci´on.
3. Probabilidad de cruce.

Los datos Toy, ver Tabla 3.3, se utilizan para la calibraci´on, sin embargo se debe mencionar que cada problema tiene caracter´ısticas que los hace diferentes a cual­quier otro y los par´ametros podr´ıa variar en su valor, por este motivo lo que se presenta a continuaci´on es un solo un ejemplo del procedimiento para ajustar los valores requeridos.

Se ha creado unprocedimiento, ver Programa 8, con elque se repite30 veces el AG para diferentes valores de los par´ametros a calibrar. Esto se lo realiz´o un n´umero grandede veces, sin embargo se muestran en losgr´aﬁcos siguientes aquellas corridas quehicieronque setomar´a unadecisi´on sobrelospar´ametros a utilizarpara encon­trar soluciones en nuestro problema de prueba.

Las Tablas 4.1, 4.2 y 4.3, contienen informaci´on de:

1. N´umero de corridas.
2. Probabilidad de cruce y/o mutaci´on.
3. Tama˜no de muestra.
4. Porcentaje de individuos de igual ﬁtness para criterio de parada.

Programa 8 “pruebas.m”: Genera continuas corridas del AG

x=[10 20 30 40 50 60 70 80 90 100]; suma=1; for i=1:length(x)

for j=1:30 suma [tt iteracion t]=agcbctt(G,C,R,CR,UC,RC,x(i),0.8,0.2); [ffitnessv ffitness]=fitnessvector(tt,G,R,UC,C); ffitnessvec(i,j)=ffitness; iteracionvec(i,j)=iteracion; tvec(i,j)=t;

end promfitness(i,1)=round(mean(ffitnessvec(i,:))); promiteracion(i,1)=round(mean(iteracionvec(i,:))); promt(i,1)=round(mean(tvec(i,:))); suma=suma+1;

end promfitness promiteracion promt

1. Valor de ﬁtness promedio en las 30 corridas para cada valor ﬁjo.
2. N´umero de iteraciones promedio en las 30 para cada valor ﬁjo.
3. Tiempodeejecuci´onpromedio(segundos) enlas30paracadavalor ﬁjo.
4. Promedios de valores encontrados.
5. Desviaci´on est´andar de valores encontrados.

La Tabla 4.1 muestra valores medios de las repeticiones de la funci´on “agcbctt” paralos valoresden =10,20,..., 100.EnlaFigura4.1 sepuede observarque cuan­dose ﬁjanlosvaloresdeprobabilidadde cruce y mutaci´on el AG parece encontrar soluciones ´optimas. El n´umero de iteraciones que se necesitan para que el AG en­cuentreunasoluci´on(nonecesariamentela ´optima) deacuerdoalaFigura4.2 se vuelve estable, entre22y26,para valoresde n iguala30,pero como se menciona an­tes, los cromosomas ´optimos se empiezan a encontrar apartirde n =50. La Figura

4.3 da una tendencia lineal de crecimiento mientras el tama˜nodela poblaci´on crece. Las pruebas para este ejemplo han mostrado que tiempos aceptables del algoritmo podr´ıan estar entre30y 50 segundos,que en elgr´aﬁco se muestranpara n entre 40 y 60.

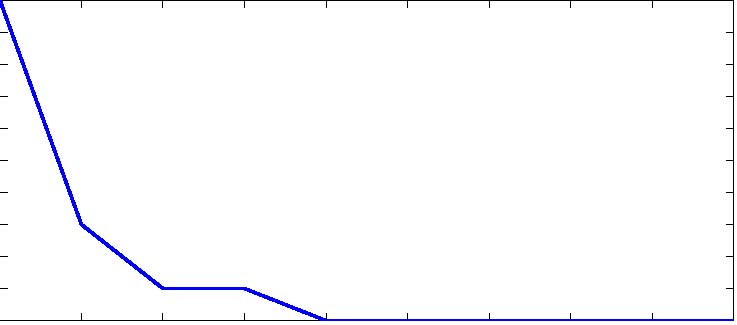
Los valores ﬁjos de las probabilidades de cruce y mutaci´on, se obtuvieron luego de analizar de forma conjunta los gr´aﬁcos para variaciones en estos par´ametros.

Tabla4.1:Experimento computacional(Tama˜nodePoblaci´on)



***Tamaño de Población vs Fitness***

10 9 8 7 6 5 4 3 2 1 0



Fitness

10 20 30 40 50 60 70 80 90100

Tamaño de Población

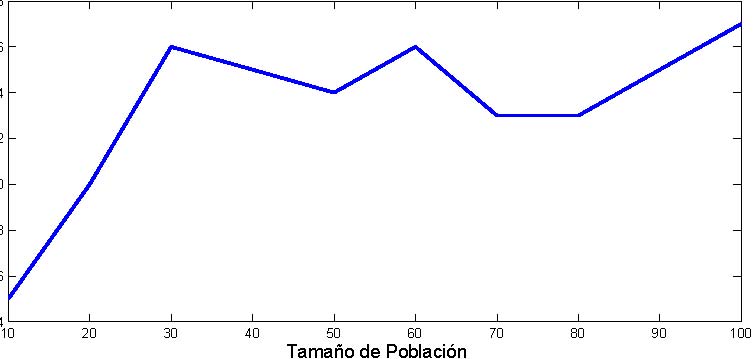
Figura 4.1: Tama˜no de Poblaci´on vs Fitness, variaci´on de poblaci´on

***Tamaño de Población vs Número de Iteraciones***

28 26

10 20 30 40 50 60 70 80 90100

Tamaño de Población



Número de Iteraciones24 22 20 18 16 14

***Tamaño de Población VS de Tiempo de Ejecución***

80

70

60

50

40

30

20

10

0

Tamaño de la Poblacón

Figura 4.3: Tama˜no de Poblaci´on vs Tiempo de Ejecuci´on, variaci´on de poblaci´on

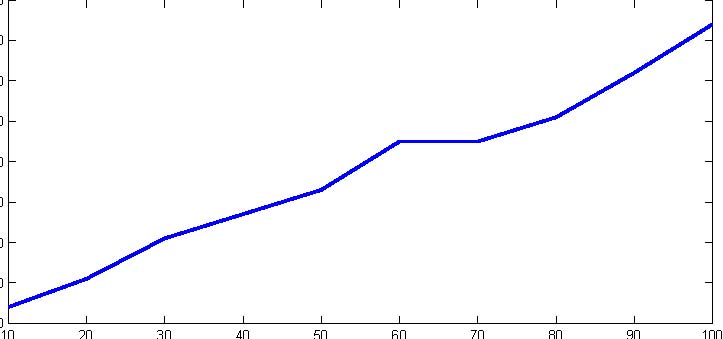
EnlaTabla4.2sehahecho variarlaprobabilidadde cruce en valoresde probcruce = 0,1, 0,2,..., 1, y se puede observar en la Figura 4.4, que el ´

optimo se encuentra a partir del valor de probabilidad de cruce de 0,5, sin embargo, un an´alisis m´as deta­llado mostr´o que el valor de este par´ametro podr´ıa estar entre 0,6y 0,8. Para estas probabilidades de cruce, el promedio del n´umero de iteraciones que se necesitaron, est´a entre 23 y 27, ver Figura 4.5 y su tiempo de ejecuci´on promedio entre 40 y 50 segundos como lo presenta la Figura 4.6.

Tabla4.2:Experimento computacional(ProbabilidaddeCruce)

Tiempo de Ejecución

10 20 30 40 50 60 70 80 90100



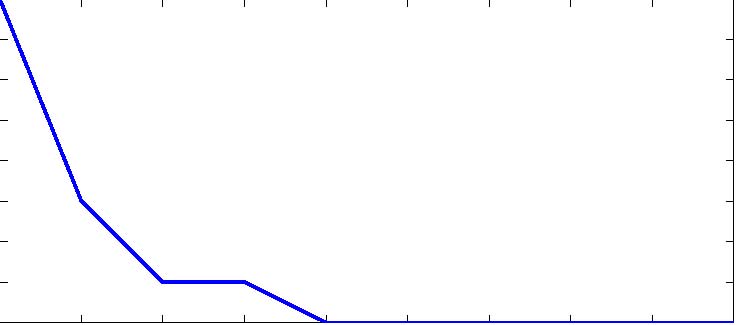
|  |  |  |  |
| --- | --- | --- | --- |
| Par´ametros Corridas c/pob 30 n 60 Prob. mutaci´on 0.4 % de igual ﬁtness 0.8 probcruce Fitness 0.1 8 0.2 3 0.3 1 0.4 1 0.5 0 0.6 0 0.7 0 0.8 0 0.9 0 1 0 | | Iteraciones 23 26 27 26 26 27 24 23 24 24 | Tiempo ejecuci´on (seg) 16 21 27 31 36 46 45 47 56 62 |

Un an´alisis similar alosdos anteriores se realiza enlaTabla4.3y enlasFiguras4.7,

4.8 y 4.9, donde se establecen valores de probabilidad de mutaci´on mayores de 0,5 para un n´umero de iteraciones promedio entre 0,3y0,7, y un tiempo de ejecuci´on aproximado de 50 segundos.

***Probabilidad Cruce vs Fitness***

8 7 6 5 4 3 2 1 0



Fitness

0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

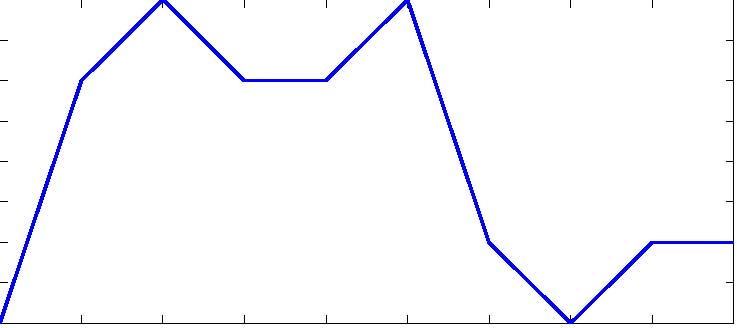
Probabilidad Cruce

Figura 4.4: Probabilidad de Cruce vs Fitness, variaci´on de probabilidad de cruce

***Probabilidad Cruce vs Número de Iteraciones***

27

26.5



Número de Iteraciones

26

25.5 25

24.5 24

23.5 23

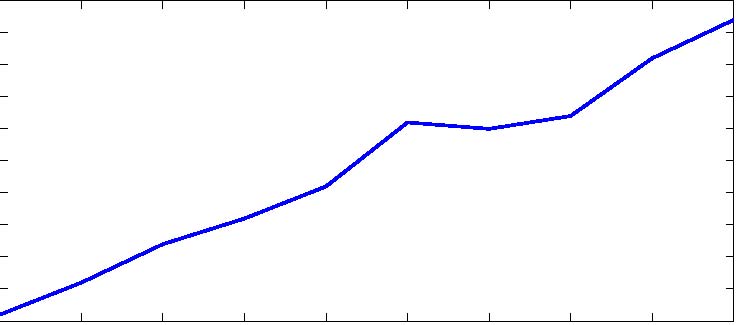
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Probabilidad Cruce

Figura 4.5: Probabilidad de Cruce vs N´umero de Iteraciones, variaci´on de probabi­lidad de cruce

Tiempo de Ejecución

50 45 40 35 30 25 20 15



0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

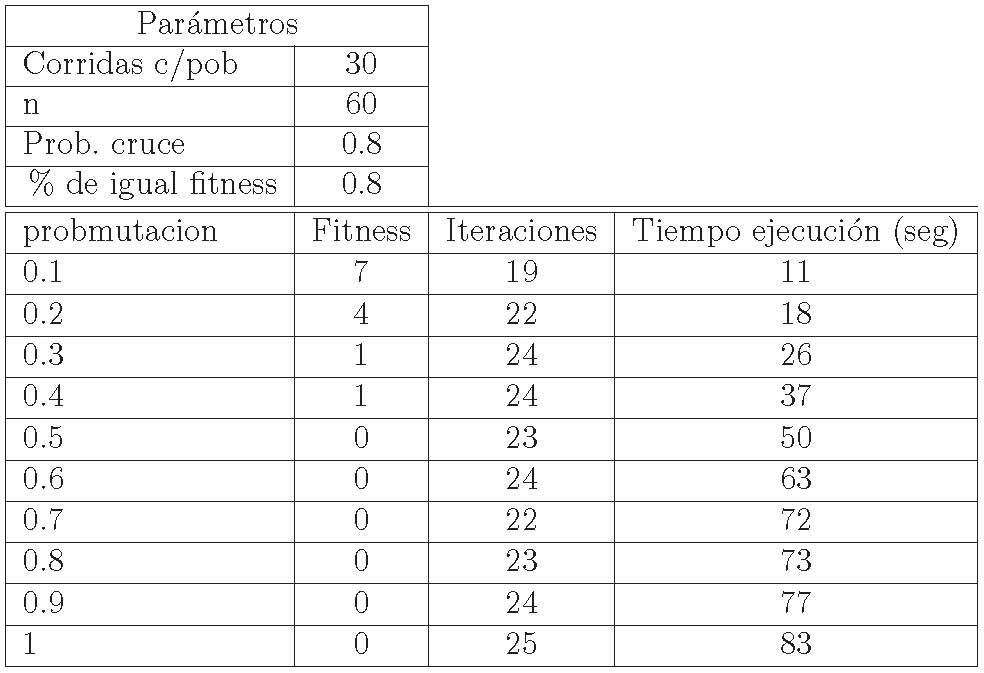
Probabilidad Cruce

***Probabilidad Cruce vs Tiempo de Ejecución***

65 60 55

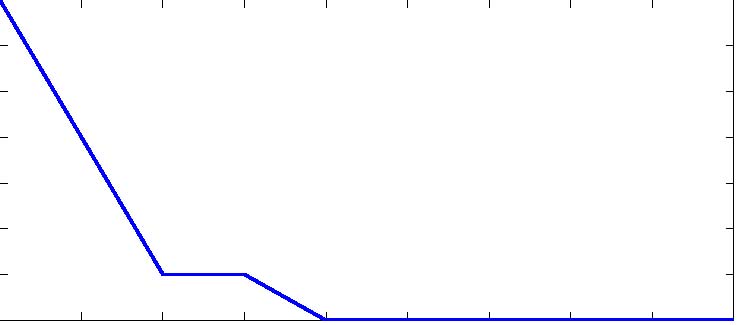
Figura 4.6: Probabilidad de Cruce vs Tiempos de Ejecuci´on, variaci´on de probabi­lidad de cruce

Tabla4.3:Experimento computacional(ProbabilidaddeMutaci´on)



***Probabilidad de Mutación vs Fitness***

7 6 5 4 3 2 1 0



Fitness

0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Probabilidad de Mutación

Figura 4.7: Probabilidad de Mutaci´on vs Fitness, variaci´on de probabilidad de mu­taci´on

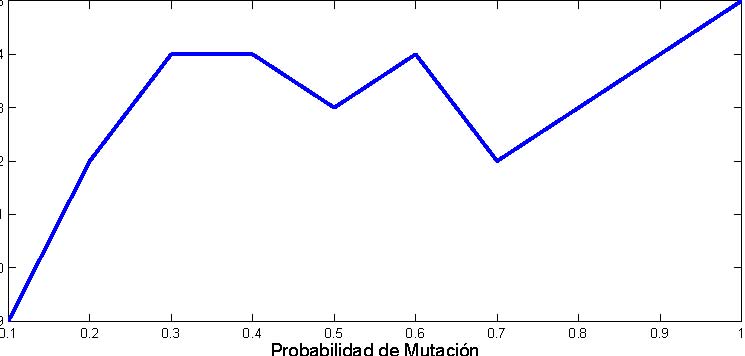
***Probabilidad de Mutación vs Número de Iteraciones***

25

24

0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Probabilidad de Mutación



Número de Iteraciones23 22 21 20 19

Figura 4.8: Probabilidad de Mutaci´on vs N´umero de Iteraciones, variaci´on de pro­babilidad de mutaci´on

***Probabilidad de Mutación vs Tiempo de Ejecución***

90 80 70 60 50 40 30 20 10

Probabilidad de Mutación

Figura 4.9: Probabilidad de Mutaci´on vs Tiempos de Ejecuci´on, variaci´on de pro­babilidad de mutaci´on

4.2. Soluciones al problema de Calendarizaci´on

4.2.1. Base de datos TOY Se ha ejecutado “agcbctt” parala base de datos Toy del archivo toy.txt, ver Tabla 3.3, y se han obtenido los calendarios de las Figuras 4.10 y 4.11 que cumplen con todas las restricciones impuestas en la formulaci´on UD4, ver Tabla 3.2, en su ver­si´on ectt. Lo par´ametros utilizados para este ejemplo fueron: Tama˜no de la poblaci´on: 60

Probabilidad de Cruce: 0.7 Probabilidad de Mutaci´on: 0.4

Tiempo de Ejecución

0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

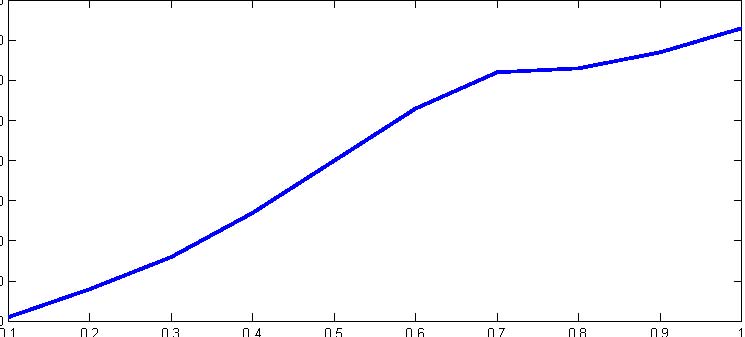


Figura 4.10: Calendario para el Curr´ıculum 1, Toy En la ﬁgura 4.12 se muestra la funci´on de evaluaci´on de la soluci´on presentada.

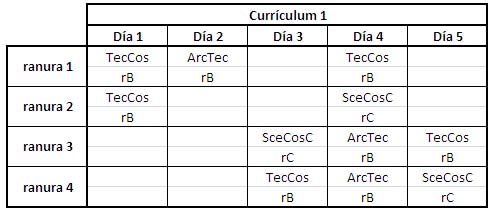
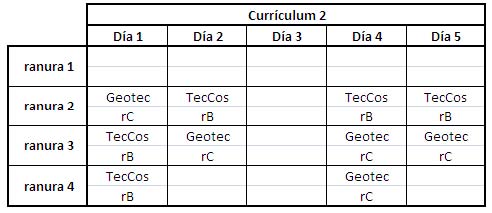


Figura 4.11: Calendario para el Curr´ıculum 2, Toy



>> [ffitnessv ffitness]=fitnessvector(tt,G,R,UC,C) ffitnessv =

00000000 ffitness =

0

Figura 4.12: Aplicaci´on de la funci´on ﬁtness a la base de datos Toy

El n´umero de iteraciones que le tom´o al AG encontrar una soluci´on fue 22 y el tiempo de ejecuci´on de 19 segundos.

Se puede encontrar m´as de una soluci´on para este problema como la presentada en lap´agina web del SaTT: Scheduling and Timetabling Group, ver Secci´on3.2.1yque se muestra en el Ap´endice B.

La soluci´on en MatlabRon 40 × 8, ver Figura 4.13. El

® es una matriz de dimensi´calendariopara el curr´ıculum 1y2se muestrande color azuly rojo respectivamente.

4.2.2. Un ejemplo con tres curr´ıculums

Se corri´o un ejemplo adicional con tres curr´ıculums, ver Tabla C.1 en el Ap´endice C,los resultadospuedenleerse enlasFiguras4.14,4.15y 4.16, adem´asla salidatesi de la funci´on de ﬁtness, con su valor ´optimo se encuentra en la Figura 4.17.

Lo par´ametros utilizados para este ejemplo fueron:

Tama˜no de la poblaci´on: 60

Probabilidad de Cruce: 0.8

Probabilidad de Mutaci´on: 0.4

Figura 4.13: Calendario en formato matriz, para la soluci´on de Toy

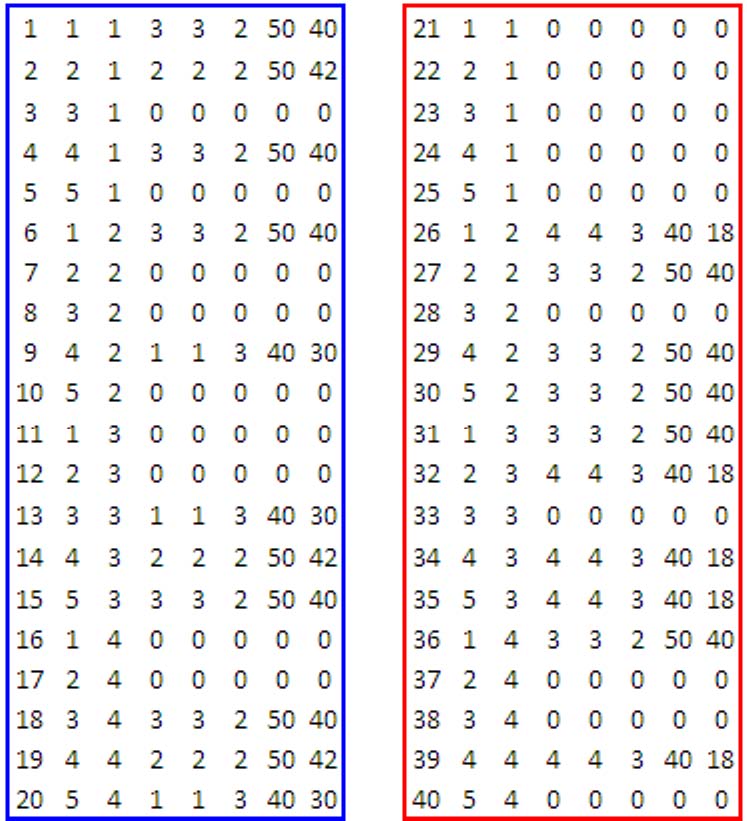


Figura 4.14: Calendario para el Curr´ıculum 1, ejemplo 2

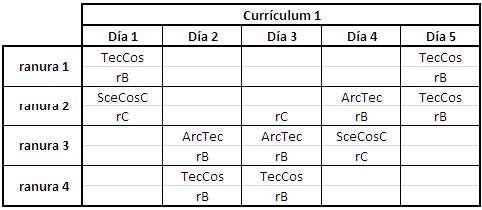


Figura 4.15: Calendario para el Curr´ıculum 2, ejemplo 2

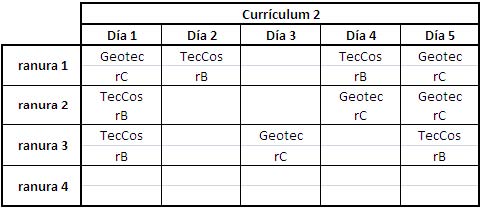
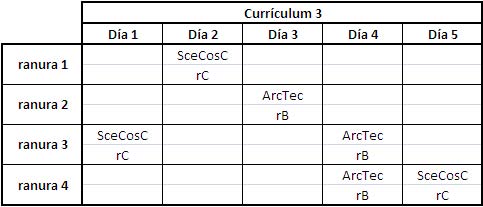


Figura 4.16: Calendario para el Curr´ıculum 3, ejemplo 2



>> [ffitnessv ffitness]=fitnessvector(tt,G,R,UC,C) ffitnessv =

00000000 ffitness =

0

Figura 4.17: Aplicaci´on de la funci´on ﬁtness a la base de datos del ejemplo 2

55

Cap´ıtulo 5

Conclusiones y Recomendaciones

5.1. Conclusiones

1. LosM´etodosMetaheur´ısticosparaelproblemadeCalendarizaci´on,espec´ıﬁca­mente los del sector educativo, son ampliamente utilizados, aunque se puede encontrar en la literatura aplicaciones que los combinan con m´etodos exactos tal como sepuede ver en Oliveri[15]. Entre lasMetaheur´ısticas m´as utilizadas est´an las que contienen memoria en sus procedimientos y guardan informa­ci´on de soluciones ya antes visitadas como los m´etodos Tab´u, Jin-Kao[7]. Se ha podido encontrar tambi´en la aplicaci´on de algoritmos evolutivos para este problema, muchas vecesdetipoh´ıbrido,Ki-Seok[19], as´ı comodel Recocido Simulado que ha sido utilizado en la soluci´on de los datos Toy por Andrea Schaerf de la Universidad de Udine, Italia.
2. Lainclusi´onderestriccionesadicionalesnoesunproblemahablandoent´ermi­nos computacionales. La diﬁcultad solamente radica en el tiempo que tome programar el algoritmo que se dise˜ne para este prop´osito. En estainvestiga­ci´on,por ejemplo,las restricciones Curr´ıculum Compacto y Curr´ıculum Com­pacto versi´on 2 pudieroninclu´ırse una o ambas enlafunci´onde evaluaci´on.La formulaci´on UD no considera a Curr´ıculum Compacto versi´on 1. Un proble­ma que suele presentarse es la de disponibilidad de horarios por parte de los profesores,peropuede relacionarse con elgrupode restriccionesde Nodisponi­bilidad presentada como UNAVAILABILITY CONSTRAINTS en el formato de datos de entrada, ver Tabla 3.3.
3. El algoritmo de optimizaci´on sirve de base para que se dise˜ne y programe una interfaz gr´aﬁca para usuarios donde se muestre los recursos asignados y disponibles, con restricciones respectivas para reales, y que pueda mostrar calendarios ﬁltrados para profesores, por materias y por aulas.
4. Para problemas de calendarizaci´on grandes el tiempo de ejecuci´on puede in­crementarse a valores no aceptables, es por esto que la aleatorizaci´on para construir los cromosoma de la poblaci´on inicial podr´ıa sustituirse por alg´un m´etodo sistem´atico opor otraparte esposiblegenerar soluciones notanbue­nas mediante el uso de laprogramaci´on lineal con restricciones relajadas. Este

tipo de mezcla de t´ecnicas de soluci´on es una alternativa digna de considerar, un ejemplo en problemas de calendarizaci´on deportiva se puede consultar en

D. Oliveri[15].

5. Luego de probar en algunos ejemplos el AG dise˜nado, se concluye que las calibraci´on de par´ametros debe realizarse para cada problema particular, no solamente por el tama˜no del mismo, sino por el tipo de formulaci´on que se considere. Las diferentes restricciones imprimen diﬁcultades distintas en cada caso.

5.2. Recomendaciones

1. Una de las cosas en las que se considera que un Algoritmo gen´etico puede mejorar, es en la administraci´on de la memoria, para guardar soluciones ya consideradas, tal como lo hacen los m´etodos Tab´u, una referencia de c´omo lograrlopueden encontrarseenlasaplicacionesquepodr´ıandarsedel Scatter Search,Glover[6], m´etodoque adem´asde(en algunade sus versiones) con­siderar memoria, considera tama˜nos de poblacion m´as peque˜nos enfoc´andose fuertemente enla calidadde sus miembros,Laguna[13].
2. En esta implementaci´on del procedimiento general del AG se considera un gran n´umerode restricciones(8 entotal,4durasy 4 suaves) que sin embargo pueden ampliarse.Por ejemplo, una restricci´onque no es considerada eslade distancia recorrida por los estudiantes cuando deben cambiar de aula entre clase y clase. El ediﬁcio donde se encuentran estos salones pueden ubicarse lejos una de la otra, de tal forma que se deber´ıa tomar en cuenta que a un estudiantele tomar´a un tiempo mayorque ceropasar a otro curso en este caso y no pueden programarse lecturas entre estos cambios.
3. La optimizaci´on de la programaci´on de las actividades acad´emicas de la ES­POL, pienso deber´ıa ser considerada con mayor ´enfasis por las instancias res­pectivas,al ser un conjuntode accionesquedeﬁnitivamentedisminuyelashoras de trabajode acoplamientodehorariosy aulas, as´ı como reducelosgastosque podr´ıa demandar una planiﬁcaci´on elemental de los recursos.
4. Se Sugiere estudiar la variante del problema de calendarizaci´on ETT, ver Secci´on 1.1, por el uso continuo que se podr´ıa dar en el ICM y en todas las unidades acad´emicas de la ESPOL. La aplicaci´on de un Algortimo G´enetico para este caso se vuelve un poco m´as sencilla debido a que los estudiantes pueden ser considerados como entidades completas de estudio.
5. Unestudiosobreel n´umerodeestudiantesquetomanlosbusesdelaESPOL en diferenteshorasdeld´ıa, ser´ıaimportantepara modelar una nueva restricci´on. Un problema que se identiﬁca es el retraso de estudiantes a la primera hora, yunaprogramaci´onde clasesquetomeenconsideraci´on esta situaci´onpodr´ıa ayudar en algo este inconveniente.

Ap´endice A Programas adicionales

Programa 9 “gcrom.m”: Genera un cromosoma

function cromosoma=gcrom(G,C,CR,RC,R,UC) clc; days=G(3,1); periods=G(4,1); rooms=G(2,1); courses=G(1,1); curricula=G(5,1); le=days\*periods\*curricula; de=le/curricula; sr=size(R); for i=1:le

cromosoma(i,1)=i; end sum=0; for j=1:le/days

for i=1:days

cromosoma(i+sum,2)=i; end sum=sum+days;

end sumdos=0; sumtres=0; for k=0:(curricula-1)

for i=1:periods for j=1:days

cromosoma(j+sumtres+sumcuatro,3)=1+sumdos; end sumdos=sumdos+1; sumtres=sumtres+days;

end

sumdos=0; end cromosoma=generalecyprof(cromosoma,C,CR,RC,R,days,periods,curricula); mejoraroom;

Programa 10 “roomycap.m”: Ubica un aula y su capacidad

ss=size(R); rr=ceil(ss(1)\*rand); fr=find(rr==RC(:,2)); if isempty(fr)==0

while ismember(cromosoma(r+days\*periods\*(i-1),4),RC(fr,1))==1 rr=ceil(ss(1)\*rand); fr=find(rr==RC(:,2));

end cromosoma(r+days\*periods\*(i-1),6)=R(rr,1); cromosoma(r+days\*periods\*(i-1),7)=R(rr,2);

else cromosoma(r+days\*periods\*(i-1),6)=R(rr,1); cromosoma(r+days\*periods\*(i-1),7)=R(rr,2);

end

Programa 11 “mejoraroom.m”: Mejora un cromosoma asignando nuevas aulas

suma=0; for i=1:curricula

for i=1:courses posm=[]; posm=find(cromosoma((1+suma:de+suma),4)==i)+suma; for h=1:length(posm)

rr=ceil(sr(1)\*rand); cromosomamdos=cromosoma; cromosomados(posm(h),6)=R(rr,1); cromosomados(posm(h),7)=R(rr,2); fitmutadodos=fitness(cromosomamdos,G,R,UC,C); fitnormaldos=fitness(cromosoma,G,R,UC,C); if fitmutadodos<fitnormaldos

cromosoma(posm(h),6)=R(rr,1); cromosoma(posm(h),7)=R(rr,2); end

end end suma=suma+de;

end

Programa 12 “generalecyprof.m”: Ubica una lectura y un profesor

function cromosoma=generalecyprof(cromosoma,C,CR,RC,R,days,periods, curricula) s=size(CR); r=[]; for i=1:curricula

memo=[];

for j=3:s(2) curso=find(CR(i,j)==C(:,1)); if isempty(find(CR(i,j)==C(:,1), 1))==0

lec=C(curso,3);

for k=1:lec %r=round(rand\*(days\*periods-1)+1); r=ceil(days\*periods\*rand); if ismember(r,memo)

while ismember(r,memo) r=ceil(days\*periods\*rand);

end cromosoma(r+days\*periods\*(i-1),4)=curso; cromosoma(r+days\*periods\*(i-1),5)=C(curso,2); mem(k)=r; roomycap; cromosoma(r+days\*periods\*(i-1),8)=C(curso,5);

else cromosoma(r+days\*periods\*(i-1),4)=curso; cromosoma(r+days\*periods\*(i-1),5)=C(curso,2); mem(k)=r; roomycap; cromosoma(r+days\*periods\*(i-1),8)=C(curso,5);

end memo=union(memo,mem); end end end end

Programa 13 “roomoccupancy.m”: Procedimiento para Aulas ocupadas

s=size(R); dp=le/curricula; cur=[]; suma=0; for i=1:dp

suma=0;

for j=1:curricula cur(i,j)=cromosoma(i+suma,6); suma=suma+dp;

end if sum(cur(i,:))==0 cur(i,curricula+1)=0; else

for k=1:s(1) c=find(cur(i,:)==k); if c>=0

prev(k)=length(c)-1; else

prev(k)=0; end cur(i,curricula+1)=sum(prev);

end

end end froomoccupancy=sum(cur(:,curricula+1));

Programa 14 “conﬂicts.m”: Procedimiento para Conﬂictos

dp=le/curricula; cur=[]; suma=0; for i=1:dp

suma=0;

for j=1:curricula cur(i,j)=cromosoma(i+suma,5); suma=suma+dp;

end if sum(cur(i,:))==0 cur(i,curricula+1)=0;

else c=find(cur(i,:)~=0); d=[]; for k=1:length(c)

d(k)=cur(i,c(k)); dset=unique(d); cur(i,curricula+1)=length(c)-length(d);

end

end end fconflicts=sum(cur(:,curricula+1));

Programa 15 “uconstraints.m”: Procedimiento para Disponibilidad

s=size(UC);

for i=1:s(1) uco=find(cromosoma(:,2)==UC(i,2) & cromosoma(:,3)==UC(i,3)); for j=1:length(uco)

if UC(i,1)==cromosoma(uco(j),4) cromosoma(uco(j),9)=1; else cromosoma(uco(j),9)=0; end

end end %cromosoma; fuconstraints=sum(cromosoma(:,9));

Programa 16 “roomcapacity.m”: Procedimiento para Capacidad de aulas

cromosoma(:,9)=[]; for i=1:le if cromosoma(i,7)<cromosoma(i,8) cromosoma(i,9)=cromosoma(i,8)-cromosoma(i,7); else cromosoma(i,9)=0;

end end cromosoma; froomcapacity=sum(cromosoma(:,9));

Programa 17 “minwdays.m”: Procedimiento para D´ıas de trabajo m´ınimo

cromosoma(:,9)=[]; s=size(C); de=le/curricula; suma=0; for j=1:curricula

for k=1:s(1) for i=1:de if cromosoma(i+suma,4)==C(k,1) cc(i,k)=cromosoma(i,2); else cc(i,k)=0;

end end if length(setdiff(unique(cc(:,k)),0))<C(k,4)

&& sum(unique(cc(:,k)))~=0 di=C(k,4)-length(setdiff(unique(cc(:,k)),0)); vec(k,j)=di;

end if sum(unique(cc(:,k)))==0 vec(k,j)=0;

end end cc=[]; suma=suma+dp;

end fminwdays=sum(sum(vec));

Programa 18 “ccompactness.m”: Procedimiento para Curr´ıculum compacto

de=le/curricula; ccomp=0; suma=0; for j=1:curricula

pos=[]; vec=[]; for k=1:days

pos=find(cromosoma(1:de,2)==k)+suma; for i=1:periods if cromosoma(pos(i),4)~=0 vec(i,k)=cromosoma(pos(i),3); else vec(i,k)=0;

end end f=find(vec(:,k)~=0); if length(f)>1

for m=1:length(f)-1 if f(m+1)-f(m)>1 ccomp=ccomp+1; end end

end end suma=suma+de;

end fccompactness=ccomp;

Programa 19 “rstability.m”: Procedimiento para Estabilizaci´on de aulas

de=le/curricula;

suma=0;

sumados=0;

for j=1:curricula vec=[]; for k=1:courses pos=find(cromosoma(1+suma:de+suma,4)==k)+suma; if isempty(pos)==0 for h=1:length(pos) vec(h,k)=cromosoma(pos(h),6); end rs=length(unique(vec(:,k)))-1; sumados=sumados+rs; end end

suma=suma+de;

end

frstability=sumados;

Programa 20 “cruce.m” Parte I: Proceso de cruce de un par de cromosomas

function [hijo hija]=cruce(padre,madre,G) days=G(3,1); periods=G(4,1); rooms=G(2,1); courses=G(1,1); curricula=G(5,1); le=days\*periods\*curricula; de=le/curricula; hijo=padre; hijo(:,4:8)=0; hija=hijo; suma=0; for j=1:curricula

for i=1:courses posp=[]; posm=[]; posp=find(padre((1+suma:de+suma),4)==i)+suma; posm=find(madre((1+suma:de+suma),4)==i)+suma; for k=1:length(posp)

fc=[]; r=rand; if r>0.5

if hijo(posp(k),4)==0 hijo(posp(k),4:8)=padre(posp(k),4:8); if hija(posm(k),4)==0

hija(posm(k),4:8)=madre(posm(k),4:8);

else fc=[]; fc=find(hija(1+suma:de+suma,4)==0)+suma; pp=ceil(length(fc)\*rand()); hija(fc(pp),4:8)=madre(posm(k),4:8);

end else

if hijo(posm(k),4)==0 hijo(posm(k),4:8)=padre(posp(k),4:8); if hija(posp(k),4)==0

hija(posp(k),4:8)=madre(posm(k),4:8);

Programa 21 “cruce.m” Parte II: Proceso de cruce de un par de cromosomas

else fc=[]; fc=find(hija(1+suma:de+suma,4)==0)+suma; pp=ceil(length(fc)\*rand()); hija(fc(pp),4:8)=madre(posm(k),4:8);

end

else fc=[]; fc=find(hijo(1+suma:de+suma,4)==0)+suma; pp=ceil(length(fc)\*rand()); hijo(fc(pp),4:8)=padre(posp(k),4:8); if hija(fc(pp),4)==0

hija(fc(pp),4:8)=madre(posm(k),4:8);

else fc=[]; fc=find(hija(1+suma:de+suma,4)==0)+suma; pp=ceil(length(fc)\*rand()); hija(fc(pp),4:8)=madre(posm(k),4:8);

end end end else

if hijo(posm(k),4)==0 hijo(posm(k),4:8)=madre(posm(k),4:8); if hija(posp(k),4)==0

hija(posp(k),4:8)=padre(posp(k),4:8);

else fc=[]; fc=find(hija(1+suma:de+suma,4)==0)+suma; pp=ceil(length(fc)\*rand()); hija(fc(pp),4:8)=padre(posp(k),4:8);

end else

if hijo(posp(k),4)==0 hijo(posp(k),4:8)=madre(posm(k),4:8); if hija(posm(k),4)==0

hija(posm(k),4:8)=padre(posp(k),4:8);

Programa 22 “cruce.m” Parte III: Proceso de cruce de un par de cromosomas

else fc=[]; fc=find(hija(1+suma:de+suma,4)==0)+suma; pp=ceil(length(fc)\*rand()); hija(fc(pp),4:8)=padre(posp(k),4:8);

end

else fc=[]; fc=find(hijo(1+suma:de+suma,4)==0)+suma; pp=ceil(length(fc)\*rand()); hijo(fc(pp),4:8)=madre(posm(k),4:8); if hija(fc(pp),4)==0

hija(fc(pp),4:8)=padre(posp(k),4:8);

else fc=[]; fc=find(hija(1+suma:de+suma,4)==0)+suma; pp=ceil(length(fc)\*rand()); hija(fc(pp),4:8)=padre(posp(k),4:8);

end end end end

end end suma=suma+de;

end

Programa 23 “crucepoblacion.m”: Selecciona a un padre y una madre para cruzarlos y elige quien formar´a parte de la nueva generaci´on

function poblacion=crucepoblacion(poblacion,G,R,UC,C,probcruce) days=G(3,1); periods=G(4,1); rooms=G(2,1); courses=G(1,1); curricula=G(5,1); le=days\*periods\*curricula; de=le/curricula; s=size(poblacion); for i=1:s(3)

r1=ceil(s(3)\*rand); r2=ceil(s(3)\*rand); r=rand; if r<=probcruce

padre=poblacion(:,:,r1); madre=poblacion(:,:,r2); if padre==madre

poblacionbandera(:,:,i)=padre;

else [hijo hija]=cruce(padre,madre,G); fhijo=fitness(hijo,G,R,UC,C); fhija=fitness(hija,G,R,UC,C); if fhijo<=fhija

poblacionbandera(:,:,i)=hijo; else poblacionbandera(:,:,i)=hija; end end else poblacionbandera(:,:,i)=poblacion(:,:,i);

end end poblacion=poblacionbandera;

Programa 24 “mutacion.m”: Ubica un gen en un periodo disponible

function cromosoma=mutacion(cromosoma,G,R,UC,C,probmutacion) days=G(3,1); periods=G(4,1); rooms=G(2,1); courses=G(1,1); curricula=G(5,1); le=days\*periods\*curricula; de=le/curricula; sr=size(R); suma=0; for i=1:curricula

for i=1:courses posm=[]; posm=find(cromosoma((1+suma:de+suma),4)==i)+suma; for k=1:length(posm)

r=rand;

if r<probmutacion fm=[]; fm=find(cromosoma(1+suma:de+suma,4)==0)+suma; pm=ceil(length(fm)\*rand()); cromosomamut=cromosoma; cromosomamut(fm(pm),4:8)=cromosomamut(posm(k),4:8); cromosomamut(posm(k),4:8)=0; fitmutado=fitness(cromosomamut,G,R,UC,C); fitnormal=fitness(cromosoma,G,R,UC,C); if fitmutado<fitnormal

cromosoma(fm(pm),4:8)=cromosoma(posm(k),4:8); cromosoma(posm(k),4:8)=0; end end end

end suma=suma+de; end

Programa 25“mutacionpoblacion.m”:Deﬁne si un cromosoma entra alproceso de mutaci´on

function poblacion=mutacionpoblacion(poblacion,G,R,UC,C,probmutacion)

days=G(3,1);

periods=G(4,1);

curricula=G(5,1);

le=days\*periods\*curricula;

de=le/curricula;

s=size(poblacion);

for i=1:s(3) r=rand; if r<probmutacion poblacion(:,:,i)=mutacion(poblacion(:,:,i),G,R,UC,C ,probmutacion); end end

Ap´endice B Calendarios adicionales Ap´endice C

Figura B.1: Calendario para el Curr´ıculum 1, SaTT, Toy

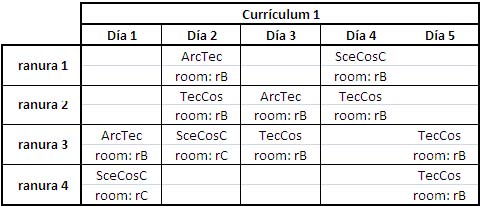
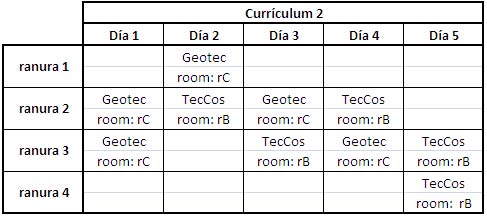


Figura B.2: Calendario para el Curr´ıculum 2, SaTT, Toy



Datos de entrada para ejemplo de tres curr´ıculums

Tabla C.1: Base de datos para ejemplo de tres curr´ıculums

Name: Toy Courses: 4 Rooms: 3 Days: 5 Periods\_per\_day: 4 Curricula: 3 Constraints: 8

COURSES: SceCosC Ocra 3 3 30 ArcTec Indaco 3 2 42 TecCos Rosa 5 4 40 Geotec Scarlatti 5 4 18

ROOMS: rA 32 rB 50 rC 40

CURRICULA: Cur1 3 SceCosC ArcTec TecCos Cur2 2 TecCos Geotec Cur3 2 SceCosC ArcTec

UNAVAILABILITY\_CONSTRAINTS: TecCos 2 0 TecCos 2 1 TecCos 3 2 TecCos 3 3 ArcTec 4 0 ArcTec 4 1 ArcTec 4 2 ArcTec 4 3

ROOM\_CONSTRAINTS: SceCosC rA Geotec rB TecCos rC

END.

Bibliograf´ıa

[1] Fabio De Cesco, Andrea Schaerf, and Luca Di Gaspero. Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, and results. 2008.

[2] Lance D. Chambers. Practical Handbook of GENETIC ALGORITHMS, Volu­men III. CRC Press LLC, 1999.

[3] Chin-Yen Chen. Using integer programming to solve the school timetabling problem at chin-min institute of technology. American Academy of Business, 2008.

[4] Adenso D´ıaz. Optimizaci?n Heur?stica y Redes Neuronales. Addison Wesley, 1996.

[5] Molina Juan Enrique. Algoritmos evolutivos para la resoluci´on de un proble­ma del tipo timetabling. Memoria para el t´ıtulo de Ingeniero en Inform?tica Aplicada, 2007.

[6] FredGloverandGaryA.Kochenberger. Handbook ofMethaheuristics. Kluwers’ International Series in Operation Research and Management Science.

[7] Jin-Kao Hao and Zhipeng Lu. Adaptive tabu search for course timetabling. ELSEVIER, 2008.

[8] J.H. Holland. Adaptation in natural and artiﬁcial systems. University of Mi-chiganPress,AnnArbor,Michigan; re-issuedbyMITPress(1992), 1975.

[9] Milena Karova, Vassil Smarkov, and Stoyan Penev. Genetic operators cros­sover and mutation in solving the tsp problem. International Conference on Computer Systems and Technologies, 2005.

[10] J.H. Kingston, R.F. Weare, E.K. Bueke, and K.S. Jackson. Automated time-tabling: The state of the art. The Computer Journal, Volume 40(9), 1997.

[11] JosephY-TLeung. HandbookofScheduling.Algorithms,Lodels andPerfomance Analysis. Chapman & Hall/CRC. Computer and Information Science Series, 2004.

´

[12] Concepci´on Maroto Alvarez and Javier Alcaraz Soria. Genetic algorithms for the resource-constrained project scheduling problem (rcpsp). Bolet´ın de Es­tad´ıstica e Investigaci?n Operativa, 2009.

[13] ManuelLagunaRafaelMart´ı. ScatterSearch:Dise?oB´asicoyEstrategiasAvan­zadas.

[14] Luis Rodrıguez Ojeda. MatlabRasicos y Programaci´on. Instituto

´® Conceptos B´de Ciencias Matem?ticas, Escuela Superior Polit?cnica del Litoral, 2007.

[15] D. Oliveri and F. Della Croce. Scheduling the italian football league: an ilp­based approach. ELSEVIER, 2004.

[16] Flores Pedro, Brau Ernesto, Moteverde Jazm?n, Salazar Norman, Figueroa Jos´e,CadenaEliseo, andLiz´arragaCaleb. Experimentos con algoritmosgen´eti­cos para resolver un problema real de programaci´on maestros-horarios-cursos.

Departamento de Matem?ticas Universidad de Sonora, Hermosillo Sonora CP 83000 M´exico.

[17] Andrea Schaerf, Fabio De Cesco, and Barry McCollum. The second interna­tional timetabling competition(itc-2007):Curriculum-based course timetabling (track 3). Technical Report, 2007.

´

[18] Aslaug S´oley Bjarnad´ottir. Solving the vehicle routing problem with genetic algorithms. Informatics and Mathematical Modelling, IMM, 2004.

[19] Ki-SeokSungandEnzheYu.Ageneticalgorithmforauniversity weekly courses timetabling problem. Blackwell Publishers, 2007.

䘀椀最甀爀愀............................................

.........................................................

.......................................................................

..........................................................

...........................................................

..............................................

.........................

.......................................................................

.................................................

........................................................

...........................................................

. . . . . . . . . .

.................

. . . . . . . . . .

.. .. .. .. .. .. .. .. ...

......................

.. .. .. .. .. .. .. .. ...

......... ........................................................................................................

............................................

..........................................................

.................................................

.................................................

.................................................

................................................

................................................