UNIVERSIDAD TECNOLÓGICA DE EL SALVADOR

FACULTAD DE INFORMÁTICA Y CIENCIAS APLICADAS

ESCUELA DE INFORMÁTICA

ESTANDARES DE PROGRAMACIÓN

UNIDAD: II

FACILITADOR: ING. EDWIN ALBERTO CALLEJAS

NOMBRE DE LA TAREA: PROPUESTA DE INVESTIGACIÓN

ESTUDIANTES:	CARNET:	PART.:
ARCE AGUIRRE, JASON ALEXANDER	25-0129-2017	100%
CASTILLO ALFARO, GABRIEL ALEJANDRO	25-3339-2005	100%
CORDERO HERNANDEZ, KATHERINE ELIZABETH	25-1461-2017	100%
JOVEL MARTINEZ, DANIEL ALEXANDER	25-2024-2010	100%
MEJIA ORELLANA, DONOVAN ERNESTO	25-1318-2014	100%
PORTILLO DELEON MIGUEL EDUARDO	25-1596-2013	100%
SALAZAR MARTINEZ, JONATHAN OSWALDO	25-6019-2016	100%

SEPTIEMBRE 10, 2020

INDICE

DESCRIPCIÓN DE LA PROPUESTA

RESOLUCIÓN DE PROBLEMAS DE HORARIOS MEDIANTE ALGORITMOS GENÉTICOS Y BÚSQUEDA HEURÍSTICA

Los algoritmos genéticos (AG) son poderosas herramientas de optimización de propósito general que modelan los principios de la evolución. A menudo, son capaces de encontrar soluciones óptimas a nivel mundial incluso en los espacios de búsqueda más complejos. Operan sobre una población de soluciones codificadas que se seleccionan de acuerdo con su calidad y luego se utilizan como base para una nueva generación de soluciones que se encuentran combinando (cruzando) o alterando (mutando) individuos actuales. Tradicionalmente, el mecanismo de búsqueda ha sido independiente del dominio, es decir, los operadores de cruce y mutación no tienen conocimiento de cuál sería una buena solución.

Una característica importante del AG es la codificación de variables que describen el problema y el método de codificación más común es transformar las variables en una cadena binaria o vector. Este proceso inicial de formulación de la población es fundamental y se reconoce como proceso de codificación.

Casi todos los institutos de educación superior tienen problemas relacionados con la programación por lo que hay que considerar muchas cosas para organizar el horario, uno de ellos es la disponibilidad de profesores y aulas físicas. No todos los profesores están disponibles en cualquier momento y algunos de ellos solo estarán disponibles en algún momento, por lo tanto, cuando se organiza el horario debe tenerse en cuenta esto junto a otras cosas como la cantidad de clases y cursos ofrecidos. El número de clases y cursos son muchos y la disponibilidad de espacio es otra cosa, el presupuesto y muchas otras.

Es obvio que, dentro del mismo semestre, todos los cursos deben programarse de manera diferente uno y otro para que el estudiante pueda tomar el curso sin ningún horario superpuesto. Todos estos cursos están registrados como grupo. Como hay cinco años de estudio, el número de grupos de cursos diferentes es un mínimo de cinco para un departamento.

Un cronograma es esencialmente un cronograma que debe adaptarse a una serie de limitaciones y las restricciones son empleadas casi universalmente por personas que se enfrentan a problemas de horarios, las restricciones se dividen a su vez casi universalmente en dos categorías: restricciones suaves y estrictas. Las restricciones duras son restricciones, de las cuales, en cualquier horario de trabajo, no habrá incumplimientos, por ejemplo: Un profesor no puede estar en dos lugares a la vez. Las restricciones blandas son restricciones que pueden romperse, pero cuyas violaciones deben minimizarse, por ejemplo: Las clases deben reservarse cerca del departamento de origen de esa clase.

Además de las limitaciones, hay una serie de excepciones que deben tenerse en cuenta al construir un sistema de programación de horarios automatizado.

Las restricciones estrictas son que las aulas no deben reservarse dos veces, cada clase debe programarse exactamente una vez y que las clases de los estudiantes no deben tener dos reservas simultáneamente, un aula debe ser lo suficientemente grande para albergar cada clase reservada, los profesores no deben reservarla simultáneamente, un profesor no debe ser reservado cuando él / ella no está disponible. Algunas clases requieren aulas particulares; algunas clases deben realizarse de forma consecutiva. Si bien las restricciones suaves son que algunos profesores prefieren programar las horas, la mayoría de los estudiantes no desean tener períodos vacíos en sus horarios, la distancia que camina un estudiante debe minimizarse, las clases deben distribuirse uniformemente durante la semana, las aulas deben reservarse cerca del departamento de origen de esa clase, no se deben reservar aulas que sean mucho más grandes que el tamaño de la clase. Además, la única restricción de excepción a considerar es que un profesor a tiempo parcial debe programarse no más de 6 unidades de tiempo, mientras que un tiempo completo es de 12 unidades de tiempo.

Los AG procesan varias soluciones simultáneamente, por lo tanto, se genera una población que tiene cromosomas llamados individuos mediante generadores pseudoaleatorios cuyos individuos representaran una solución factible. Esta es una representación del vector solución en un espacio solución y se le llama solución inicial; esto asegura que la búsqueda sea sólida e imparcial, ya que comienza desde una amplia gama de puntos en el espacio de la solución.

OBJETIVOS Y ALCANCES

OBJETIVOS

GENERAL

Solventar mediante la búsqueda heurística la resolución del problema del horario expuesto. Aunque las aulas no se incluyan en una matriz de destino con la finalidad de resolver de primera instancia los cupos de los cursos, el sistema puediese determinar qué aula se utiliza para una determinada celda en el horario según su capacidad física.

ESPECIFICOS

Evaluar si el aula es o no uno de los factores críticos para incluirse en una matriz objetivo.

Crear una matriz de destino de tres dimensiones en lugar de agregar un número de columna si el aula física es un factor crítico para solventar el problema.

ALCANCES

Hay algunas limitaciones para esta investigación. En primer lugar, cada clase paralela de un grupo de cursos que está representada en una matriz objetivo, debe programarse para cada curso definido en ese horario. En segundo lugar, solo se puede programar un curso de conferencias si hay espacio disponible consecutivamente en la matriz de destino al menos tanto como el número de unidades del curso en la respectiva aula magna de su facultad. Si ese curso tiene que dividirse en varios segmentos, entonces el nombre de ese curso debe diferenciarse en secciones. En tercer lugar, el objetivo del sistema será maximizar el número de clases exitosas que se pueden programar, de lo contrario debe definirse en consecuencia el resultado.

JUSTIFICACION

Los algoritmos genéticos son herramientas poderosas para resolver problemas de asignación y también se pueden utilizar para crear horarios de exámenes universitarios, por lo que se propone también el uso de la búsqueda heurística para resolver estos problemas de programación, por lo tanto, con un método que resuelva el problema de tabla de tiempo, mediante el uso de un algoritmo genético combinado con una búsqueda heurística, el papel del algoritmo genético seria determinar la secuencia de todos los cursos que se programarán en un grupo, mientras que el papel de la búsqueda heurística sería el de determinar los intervalos de tiempo utilizados para programar los cursos.

Por lo anterior, la investigación puede ser dividida en tres partes principales, la primera parte analiza cómo el algoritmo genético y también la búsqueda heurística pueden resolver el problema de programación. En la segunda parte se propondrá el diseño arquitectónico del sistema. En la tercera parte se mostrarán algunos experimentos y discusión después de implementar el sistema.

METODOLOGÍA DE TRABAJO

La metodología que hemos decidido aplicar para este análisis es el método descriptivo que se utiliza para recoger, organizar, resumir, presentar, analizar y generalizar los resultados de las observaciones. Este método implica la recopilación y presentación sistemática de datos para dar una idea clara de una determinada situación. Se ha implementado este tipo de metodología tomando en cuenta que es económica, efectiva y tiene un menor nivel de complejidad comparada con otras.

En este estudio descriptivo nuestro propósito como investigadores es describir las situaciones y eventos que expliquen la posibilidad de utilizar algoritmos genéticos y búsqueda heurística para solucionar problemas de programación de horario al haber consultado investigaciones antes realizadas de libros, documentos, tesis y artículos.

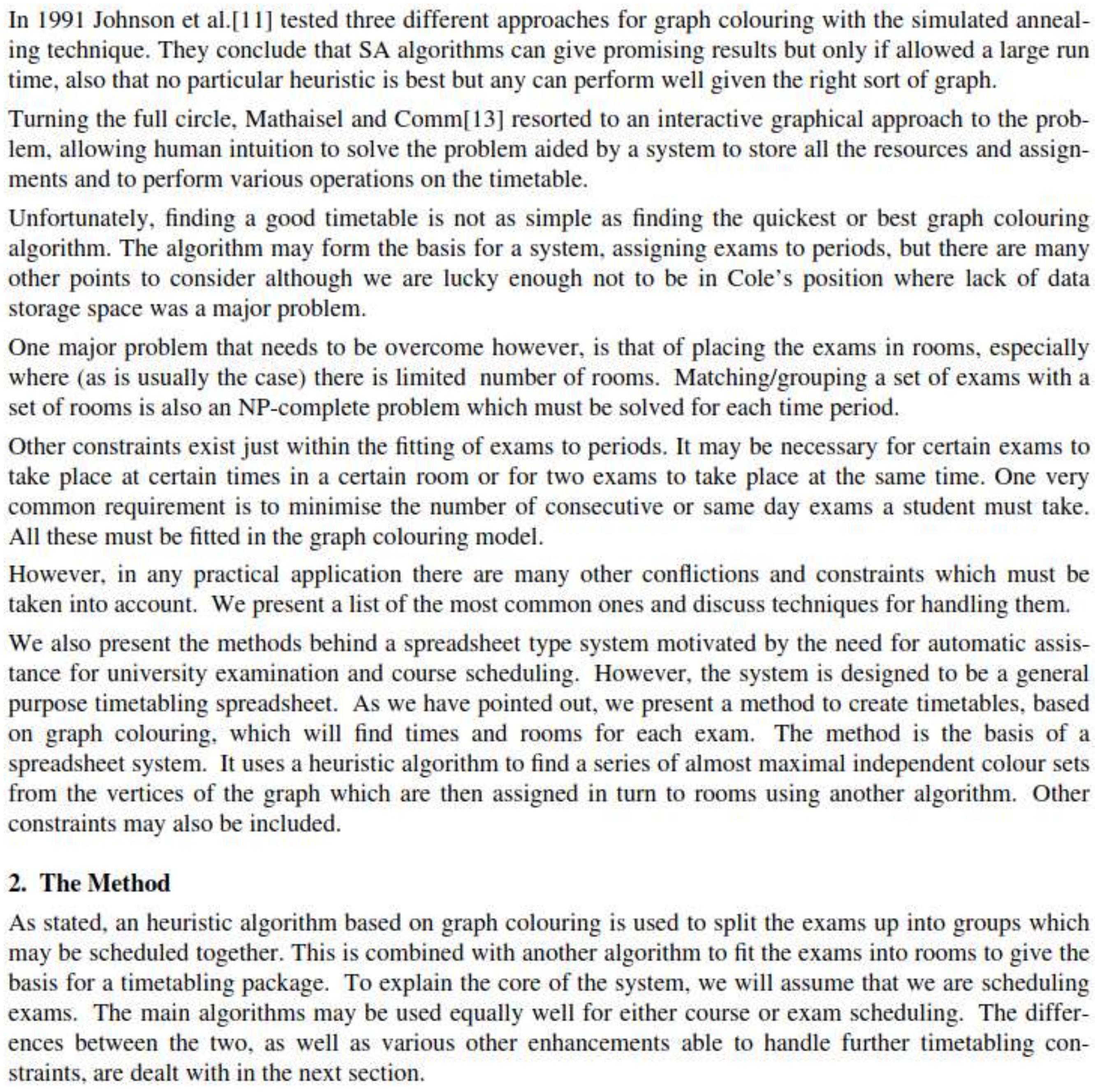
Se pretende explicar cómo es y como se manifiesta un determinado fenómeno, en este caso los algoritmos genéticos, y como se puede aplicar a una determinada situación; se necesita describir el lugar donde se quiere implementar el proyecto con pertinencia para mostrar los indicadores y las variables que deben tomarse en cuenta al trabajar con la locación, la cual en nuestro caso se trata de la Universidad Tecnológica de El Salvador.

Al utilizar este método descriptivo hemos elegido recolectar la información de fuentes confiables, verídicas y pertinentes para lograr hacer un buen estudio de los datos que queremos analizar. Se ha extraído lo más importante de los textos tratando de no perder de vista la información que se pudiera obtener de cada documento. Al observar con atención los enunciados se estudió una serie de conceptos de mucha importancia para comprender el tema a investigar.

Los datos que hemos recogido a través del método descriptivo se pueden expresar en términos cualitativos y cuantitativos, ya sea por separado o en forma conjunta. Con los datos cualitativos examinamos la naturaleza de los fenómenos y con los cuantitativos exponemos los resultados de los cálculos y las mediciones; ambos son necesarios para lograr nuestro objetivo de describir el problema que se ha presentado y mostrar claramente nuestra propuesta de solución.

PRINCIPALES FUENTES

Burke E.K., Elliman D.G. and Weare R.F. (1993a); *“A University Timetabling System Based on Graph Coloring and Constraint Manipulation”, Journal of Research on Computing in Education.* Vol. 26. issue 4
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.6.2734&rep=rep1&type=pdf



In 1991 Johnson et al.[11] tested three different approaches for graph colouring with the simulated annealing technique. They conclude that SA algorithms can give promising results but only if allowed a large run time, also that no particular heuristic is best but any can perform well given the right sort of graph.

Turning the full circle, Mathaisel and Comm[13] resorted to an interactive graphical approach to the problem, allowing human intuition to solve the problem aided by a system to store all the resources and assignments and to perform various operations on the timetable.

Unfortunately, finding a good timetable is not as simple as finding the quickest or best graph colouring algorithm. The algorithm may form the basis for a system, assigning exams to periods, but there are many other points to consider although we are lucky enough not to be in Cole’s position where lack of data storage space was a major problem.

One major problem that needs to be overcome however, is that of placing the exams in rooms, especially where (as is usually the case) there is limited number of rooms. Matching/grouping a set of exams with a set of rooms is also an NP-complete problem which must be solved for each time period.

Other constraints exist just within the fitting of exams to periods. It may be necessary for certain exams to take place at certain times in a certain room or for two exams to take place at the same time. One very common requirement is to minimise the number of consecutive or same day exams a student must take. All these must be fitted in the graph colouring model.

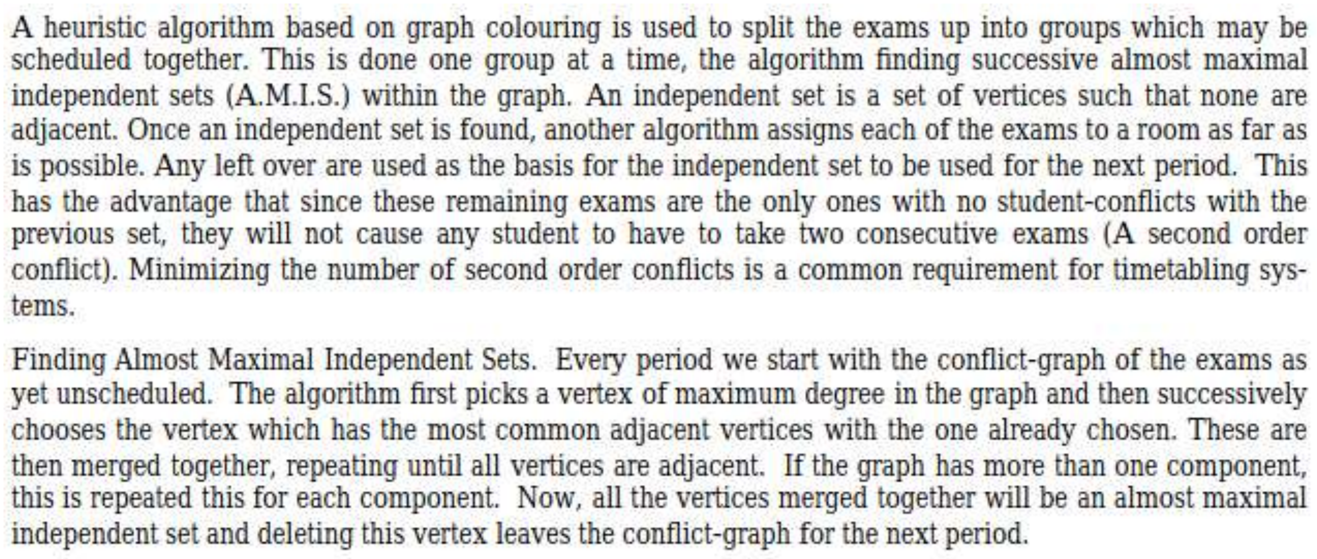
However, in any practical application there are many other conflictions and constraints which must be taken into account. We present a list of the most common ones and discuss techniques for handling them.

We also present the methods behind a spreadsheet type system motivated by the need for automatic assistance for university examination and course scheduling. However, the system is designed to be a general purpose timetabling spreadsheet. As we have pointed out, we present a method to create timetables, based on graph colouring, which will find times and rooms for each exam. The method is the basis of a spreadsheet system. It uses a heuristic algorithm to find a series of almost maximal independent colour sets from the vertices of the graph which are then assigned in turn to rooms using another algorithm. Other constraints may also be included.

2. The Method

As stated, an heuristic algorithm based on graph colouring is used to split the exams up into groups which may be scheduled together. This is combined with another algorithm to fit the exams into rooms to give the basis for a timetabling package. To explain the core of the system, we will assume that we are scheduling exams. The main algorithms may be used equally well for either course or exam scheduling. The differences between the two, as well as various other enhancements able to handle further timetabling constraints, are dealt with in the next section.

Burke E.K., Elliman D.G. and Weare R.F. (1993b); *“Automated Scheduling of University Exams”, Proceedings of I.E.E. Colloquium on “Resource Scheduling for Large Scale Planning Systems”*, Digest No. 1993/144
https://www.researchgate.net/publication/3585411_Automated_scheduling_of_university_exams



A heuristic algorithm based on graph colouring is used to split the exams up into groups which may be scheduled together. This is done one group at a time, the algorithm finding successive almost maximal independent sets (A.M.I.S.) within the graph. An independent set is a set of vertices such that none are adjacent. Once an independent set is found, another algorithm assigns each of the exams to a room as far as is possible. Any left over are used as the basis for the independent set to be used for the next period. This has the advantage that since these remaining exams are the only ones with no student-conflicts with the previous set, they will not cause any student to have to take two consecutive exams (A second order conflict). Minimizing the number of second order conflicts is a common requirement for timetabling systems.

Finding Almost Maximal Independent Sets. Every period we start with the conflict-graph of the exams as yet unscheduled. The algorithm first picks a vertex of maximum degree in the graph and then successively chooses the vertex which has the most common adjacent vertices with the one already chosen. These are then merged together, repeating until all vertices are adjacent. If the graph has more than one component, this is repeated this for each component. Now, all the vertices merged together will be an almost maximal independent set and deleting this vertex leaves the conflict-graph for the next period.

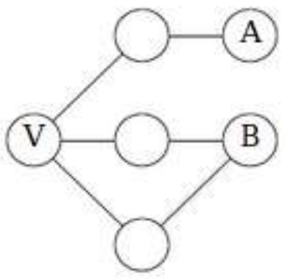


Figure 2

Burke E.K., Elliman D.G. and Weare R.F. (1994); *"A Genetic Algorithm for University Timetabling"*, AISB Workshop on Evolutionary Computing, Leeds.
https://www.researchgate.net/publication/2587988_A_Genetic_Algorithm_Based_University_Timetabling_System

solutions all of which return the same fitness value. In this case, the timetabler must find some other means of choosing the one to use.

As has been said before, the only real test of whether a timetable is of high quality is whether the institution will use it. Clearly, the more control over the search process that the timetabler has and the wider the scope of the system, the more likely it is to be accepted. Using a Graphical User Interface (G.U.I.) will allow the user to oversee the workings of the algorithm and contribute where necessary to the scheduling process. It also has the advantage that in the inevitable situation when something completely beyond the imagination of the timetable system designers occurs, it can be trivially solved.

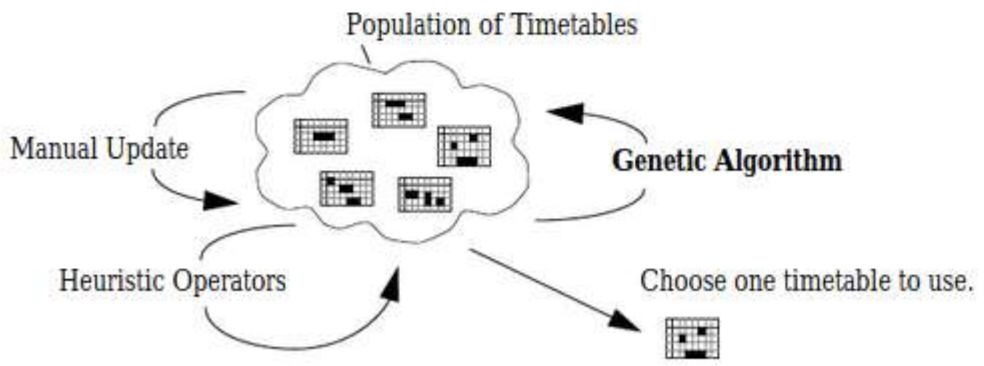


Figure 3: Searching for a Timetable

In using the system, the timetabler may use his or her own knowledge to aid the search as well as used other heuristic operators which act on the timetables to maximise various criteria while the Genetic Algorithm still provides the basis of the search and gives a choice of possible solutions. The algorithms may be left to run or the interface used to guide the timetabler to the area of greatest conflict.

1.7 Greedy Room Scheduler

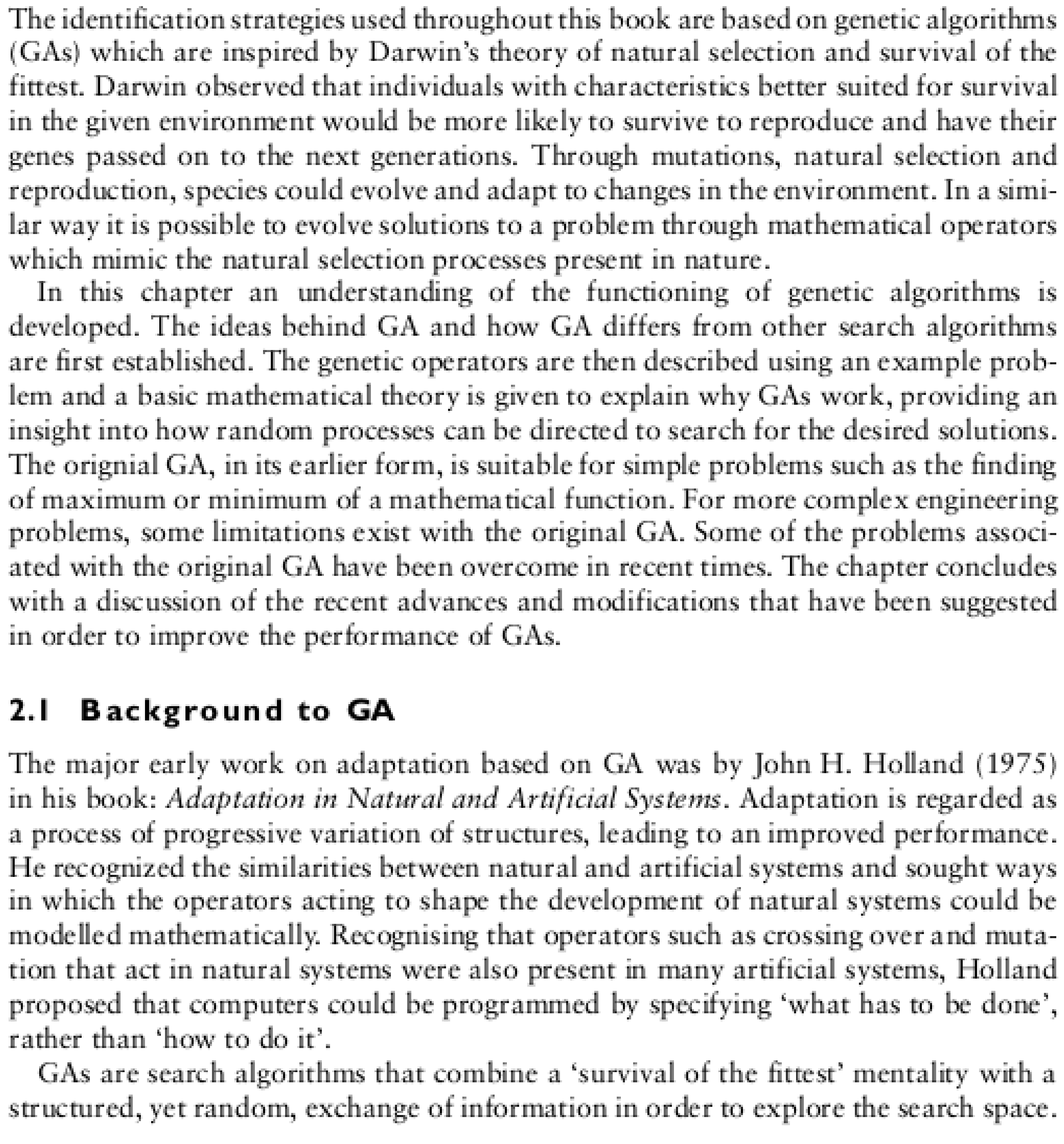
The room for each class is not determined by the GA but is determined by a knowledge-based selection procedure. With many rooms, it is assumed that any number of classes can meet at the same time, which makes it legal to have multiple copies of the same gene value on a chromosome.

The room scheduler is a greedy algorithm, sometimes called a graph-coloring scheme, that has been shown to provide good results in selection procedures [12]. A greedy algorithm begins at a specific point in the search space and works outward, selecting the first acceptable value found in the search space. It is called a greedy algorithm since it selects the first possible solution that meets the immediate, local requirements without any consideration for the complete solution. In this implementation as a room scheduler, the greedy algorithm not only selects the classrooms so no hard constraints are violated but it also minimizes the teachers' walking distances. Of course, this minimization must be considered with respect to the complete schedule optimization. Although the intent is to minimize the walking distances of the instructors, as the schedule is built, some options are removed from consideration in conjunction with some instructors. As a result, those instructors whose classes are added to the schedule later are less likely to have a minimized walking distance.

Each class is associated with an instructor, and each instructor is associated with the building where the instructor's office is located. The greedy algorithm room scheduler begins its search in the building of the instructor's office and looks for an available room at the time assigned to the class. If no room is available within the building, the next closest building is searched. This procedure continues until either an available room is located or all buildings have been searched. The room scheduler for those classes that are assigned "staff" as the instructor begins the search in the building of the department responsible for the class.

The order of the genes on the chromosome is correlated to the order of the classes on the class list. The genes, or classes, are processed sequentially. Because greedy algorithms make their selections based on local optimality, it is best to order the decisions made so the most important global decisions are made first [12]. This importance is the relative importance when using the GA automated search for timetabling and has nothing to do with the class subject. For scheduling purposes, the importance is based on the need to be assigned a room. This makes the larger classes more important than the smaller classes, since smaller classes can use larger rooms but not vice versa. The class list was therefore ordered according to expected enrollment, and the greedy algorithm finds the rooms in order of required size. No large class will be unassigned just because a smaller class has been assigned the room needed.

Davis L. (1991) *“Handbook of Genetic Algorithms”;* Van Nostrand Reinhold; ISBN-10 : 0442001738; ISBN-13 : 978-0442001735



The identification strategies used throughout this book are based on genetic algorithms (GAs) which are inspired by Darwin’s theory of natural selection and survival of the fittest. Darwin observed that individuals with characteristics better suited for survival in the given environment would be more likely to survive to reproduce and have their genes passed on to the next generations. Through mutations, natural selection and reproduction, species could evolve and adapt to changes in the environment. In a similar way it is possible to evolve solutions to a problem through mathematical operators which mimic the natural selection processes present in nature.

In this chapter an understanding of the functioning of genetic algorithms is developed. The ideas behind GA and how GA differs from other search algorithms are first established. The genetic operators are then described using an example problem and a basic mathematical theory is given to explain why GAs work, providing an insight into how random processes can be directed to search for the desired solutions. The orignial GA, in its earlier form, is suitable for simple problems such as the finding of maximum or minimum of a mathematical function. For more complex engineering problems, some limitations exist with the original GA. Some of the problems associated with the original GA have been overcome in recent times. The chapter concludes with a discussion of the recent advances and modifications that have been suggested in order to improve the performance of GAs.

2.1 Background to GA

The major early work on adaptation based on GA was by John H. Holland (1975) in his book: *Adaptation in Natural and Artificial Systems*. Adaptation is regarded as a process of progressive variation of structures, leading to an improved performance. He recognized the similarities between natural and artificial systems and sought ways in which the operators acting to shape the development of natural systems could be modelled mathematically. Recognising that operators such as crossing over and mutation that act in natural systems were also present in many artificial systems, Holland proposed that computers could be programmed by specifying ‘what has to be done’, rather than ‘how to do it’.

GAs are search algorithms that combine a ‘survival of the fittest’ mentality with a structured, yet random, exchange of information in order to explore the search space.

Erben W and Keppler J (1995): *A Genetic Algorithm Solving a Weekly Course- Timetabling Problem. In Burke E and Ross P (Eds): Lecture Notes in Computer Science 1153 Practice and Theory of Automated Timetabling First International Conference*, Edinburgh, U.K., August/September 1995, Selected Papers. New York: Springer-Verlag Berlin Heidelberg. ISSN: 0302-9743; ISBN 3-540-61794-9; pp 198-211.

2 The Weekly Course-Timetabling Problem

Our weekly course-timetabling problem involves scheduling *classes*, *teachers*, *course modules* and *rooms* to a number of *periods* (or *time slots*) in a week.

Each day of the week is divided into 6 periods (of 90 minutes' duration). There are 5 working days per week. Hence, the set P of *periods* consists of 30 elements. We denote these elements by *mon1*, *mon2*, ..., *mon6*, *tue1*, *tue2*, ..., *fri6*.

Our university offers a number of *courses* (e.g. Business Computing, Architecture, Civil Engineering, ...) in different departments, and each course lasts eight semesters. A *class* consists of all students studying a given course in the same semester. We denote by C the set of all classes. For each class $c \in C$ the (estimated) maximum number of students must be given as part of the input to our system.

T is the set of all *teachers* (lecturers, tutors, etc.). Each lecturer requires a number of free-time slots (or even one or two free days) where he or she is unavailable. These data must also be present.

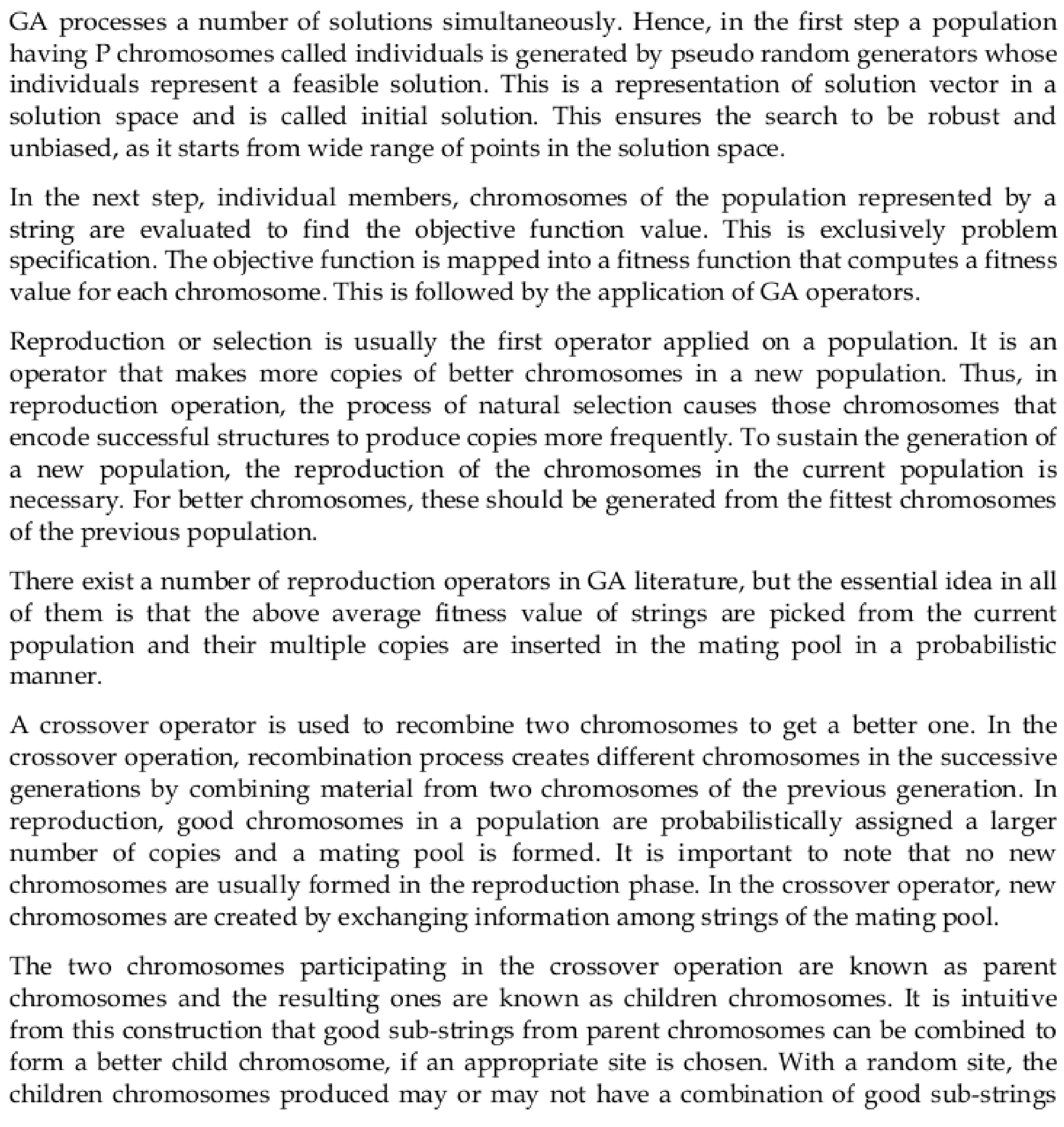
R is the set of all *rooms* available in the university. A room can be a laboratory (e.g. for chemistry, or computing) or a lecture theatre. Some rooms can only be used by the department to which they belong. Each room has a maximum capacity which must be given.

Each department offers a number of *course modules*. A course module may be taken by students of one or more classes. It may either be compulsory or optional.

A course module consists of one or more *lessons*. A lesson may be a lecture, a laboratory or a group exercise class, and it has a duration of 90 minutes. The lessons belonging to one course module may be taught by different teachers. The set of all lessons is denoted by L. For each $l \in L$ the type of the lesson, its unique teacher and the coordinate list of classes must be present.

The input data also specifies those lessons for which particular requirements exist: Some lessons must be taught in special rooms, for instance in a computing lab. Some lessons must be scheduled to more than one room (for instance, a lecture theatre and a lab; this enables a teacher to decide each week in which of the two differently equipped rooms his or her lesson should take place). Some lessons are prescheduled, i.e.

Enmund Burke, David Ellimand and Rupert Weare. (1994). *"A Genetic Algorithm Based University Timetabling System"*, August 2011; http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.2659



GA processes a number of solutions simultaneously. Hence, in the first step a population having P chromosomes called individuals is generated by pseudo random generators whose individuals represent a feasible solution. This is a representation of solution vector in a solution space and is called initial solution. This ensures the search to be robust and unbiased, as it starts from wide range of points in the solution space.

In the next step, individual members, chromosomes of the population represented by a string are evaluated to find the objective function value. This is exclusively problem specification. The objective function is mapped into a fitness function that computes a fitness value for each chromosome. This is followed by the application of GA operators.

Reproduction or selection is usually the first operator applied on a population. It is an operator that makes more copies of better chromosomes in a new population. Thus, in reproduction operation, the process of natural selection causes those chromosomes that encode successful structures to produce copies more frequently. To sustain the generation of a new population, the reproduction of the chromosomes in the current population is necessary. For better chromosomes, these should be generated from the fittest chromosomes of the previous population.

There exist a number of reproduction operators in GA literature, but the essential idea in all of them is that the above average fitness value of strings are picked from the current population and their multiple copies are inserted in the mating pool in a probabilistic manner.

A crossover operator is used to recombine two chromosomes to get a better one. In the crossover operation, recombination process creates different chromosomes in the successive generations by combining material from two chromosomes of the previous generation. In reproduction, good chromosomes in a population are probabilistically assigned a larger number of copies and a mating pool is formed. It is important to note that no new chromosomes are usually formed in the reproduction phase. In the crossover operator, new chromosomes are created by exchanging information among strings of the mating pool.

The two chromosomes participating in the crossover operation are known as parent chromosomes and the resulting ones are known as children chromosomes. It is intuitive from this construction that good sub-strings from parent chromosomes can be combined to form a better child chromosome, if an appropriate site is chosen. With a random site, the children chromosomes produced may or may not have a combination of good sub-strings

Negnevitsky, Michael. (2005). *Artificial Intelligence, A Guide to Intelligence System (2 nd),* Addison Wesley, pp. 222-245, IBN 0-321-20466-2, Harlow, England

7.5 Case study: maintenance scheduling with genetic algorithms

One of the most successful areas for GA applications includes the problem of scheduling resources. Scheduling problems are complex and difficult to solve. They are usually approached with a combination of search techniques and heuristics.

Why are scheduling problems so difficult?
First, scheduling belongs to NP-complete problems. Such problems are likely to be unmanageable and cannot be solved by combinatorial search techniques. Moreover, heuristics alone cannot guarantee the best solution.

Second, scheduling problems involve a competition for limited resources; as a result, they are complicated by many constraints. The key to the success of the GA lies in defining a fitness function that incorporates all these constraints.

The problem we discuss here is the maintenance scheduling in modern power systems. This task has to be carried out under several constraints and uncertainties, such as failures and forced outages of power equipment and delays in obtaining spare parts. The schedule often has to be revised at short notice. Human experts usually work out the maintenance scheduling by hand, and there is no guarantee that the optimum or even near-optimum schedule is produced.

A typical process of the GA development includes the following steps:

1 Specify the problem, define constraints and optimum criteria.
2 Represent the problem domain as a chromosome.
3 Define a fitness function to evaluate the chromosome's performance.
4 Construct the genetic operators.
5 Run the GA and tune its parameters.

Rich DC (1995): A Smart Genetic Algorithm for University Timetabling. In Burke E and Ross P (Eds): *Lecture Notes in Computer Science 1153 Practice and Theory of Automated Timetabling First International Conference, Edinburgh, U.K., August/September 1995*, Selected Papers. New York: Springer-Verlag Berlin Heidelberg. pp 181-197.

1.8 Greedy Time Search

Even the dynamic penalty function could not sufficiently increase the pressure to resolve all hard constraints in a minimum-resource environment. A greedy time search algorithm was implemented to resolve time conflicts quicker and to help the dynamic penalty function in difficult environments. Time conflicts are caused by the GA-evolved time assignment for a class overloading campus resources that are available at that time. A conflict is also created when the GA assigns a starting time too late in the week for the class to have the assigned number of meetings with the preferred time span allowed between meetings.

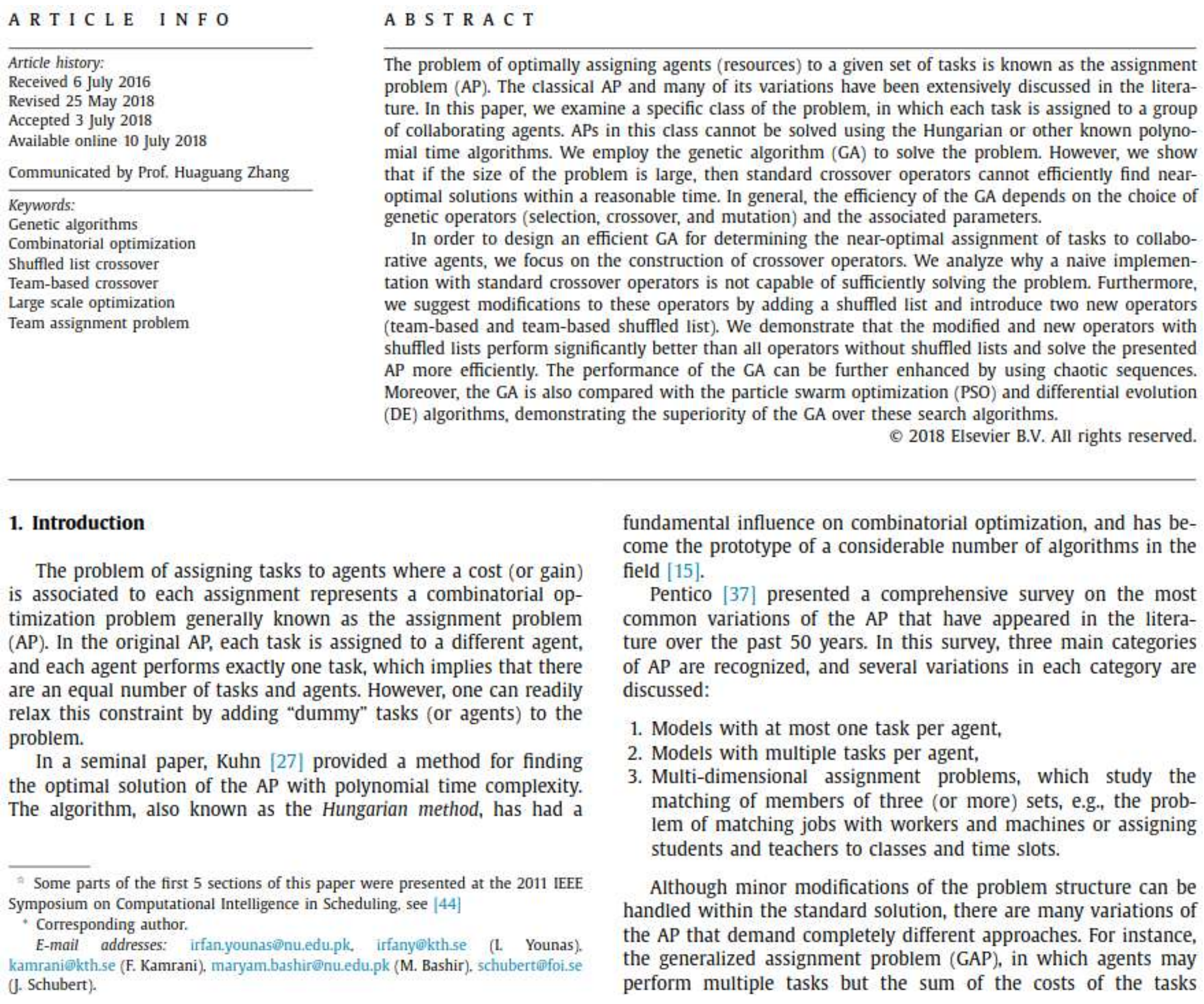
When the GA evolves a time conflict, a search begins to find a better time for that class. The search begins with the time identified as "prime time," since that time will eventually have the maximum number of classes. The first period in the week is currently used as the prime time, but the prime time could be set to any other period in the week. The greedy time search algorithm first finds a time for the class that does not result in a hard cost due to the instructor's time requirements. The greedy room scheduler then searches for a room. If no available room can be found, the greedy time search continues searching for the next available time.

The greedy time search can be turned on or off to evaluate the operation of the program with and without using it. The search can also be turned on part way through a run. For example, the program can evaluate 25% (or any other amount) of the maximum number of chromosomes to be evaluated without using the greedy time search and then turn on the search. The results of turning on the time search are very dramatic. A plot of the best fitness value versus the chromosomes evaluated shows that although the best fitness may slowly improve, when the time search is turned on, the best fitness value improves dramatically. (See Figures 2-7.)

Although a complete analysis has not yet been done, it is thought best to leave the time search off for the first part of the evolutionary process. This puts pressure on the scheduler to resolve the constraints as much as possible by juggling the assignments. It is thought that this initial juggling without the greedy time search will help in locating the best solution.

The program was also tested using the greedy time search without the dynamic penalty costs. The results were better than using only the dynamic penalty function without the greedy time search, but an acceptable schedule could not be found in a minimum-resource environment. The best results are achieved by using both the greedy time search and the dynamic penalty function.

S. Lukas, P. Yugopuspito and H. Asali, (2005); *"Solving assignment problem by genetic algorithms using Cycle Crossover"*, Universitas Pelita Harapan Computer Science Journal, Vol. 3, No. 2, Mei, pp. 87-93.



A R T I C L E   I N F O

Article history:
Received 6 July 2016
Revised 25 May 2018
Accepted 3 July 2018
Available online 10 July 2018

Communicated by Prof. Huaguang Zhang

Keywords:
Genetic algorithms
Combinatorial optimization
Shuffled list crossover
Team-based crossover
Large scale optimization
Team assignment problem

A B S T R A C T

The problem of optimally assigning agents (resources) to a given set of tasks is known as the assignment problem (AP). The classical AP and many of its variations have been extensively discussed in the literature. In this paper, we examine a specific class of the problem, in which each task is assigned to a group of collaborating agents. APs in this class cannot be solved using the Hungarian or other known polynomial time algorithms. We employ the genetic algorithm (GA) to solve the problem. However, we show that if the size of the problem is large, then standard crossover operators cannot efficiently find near-optimal solutions within a reasonable time. In general, the efficiency of the GA depends on the choice of genetic operators (selection, crossover, and mutation) and the associated parameters.

In order to design an efficient GA for determining the near-optimal assignment of tasks to collaborative agents, we focus on the construction of crossover operators. We analyze why a naive implementation with standard crossover operators is not capable of sufficiently solving the problem. Furthermore, we suggest modifications to these operators by adding a shuffled list and introduce two new operators (team-based and team-based shuffled list). We demonstrate that the modified and new operators with shuffled lists perform significantly better than all operators without shuffled lists and solve the presented AP more efficiently. The performance of the GA can be further enhanced by using chaotic sequences. Moreover, the GA is also compared with the particle swarm optimization (PSO) and differential evolution (DE) algorithms, demonstrating the superiority of the GA over these search algorithms.

1. Introduction

The problem of assigning tasks to agents where a cost (or gain) is associated to each assignment represents a combinatorial optimization problem generally known as the assignment problem (AP). In the original AP, each task is assigned to a different agent, and each agent performs exactly one task, which implies that there are an equal number of tasks and agents. However, one can readily relax this constraint by adding "dummy" tasks (or agents) to the problem.

In a seminal paper, Kuhn [27] provided a method for finding the optimal solution of the AP with polynomial time complexity. The algorithm, also known as the *Hungarian method*, has had a fundamental influence on combinatorial optimization, and has become the prototype of a considerable number of algorithms in the field [15].

Pentico [37] presented a comprehensive survey on the most common variations of the AP that have appeared in the literature over the past 50 years. In this survey, three main categories of AP are recognized, and several variations in each category are discussed:

1. Models with at most one task per agent,
2. Models with multiple tasks per agent,
3. Multi-dimensional assignment problems, which study the matching of members of three (or more) sets, e.g., the problem of matching jobs with workers and machines or assigning students and teachers to classes and time slots.

Although minor modifications of the problem structure can be handled within the standard solution, there are many variations of the AP that demand completely different approaches. For instance, the generalized assignment problem (GAP), in which agents may perform multiple tasks but the sum of the costs of the tasks

☆ Some parts of the first 5 sections of this paper were presented at the 2011 IEEE Symposium on Computational Intelligence in Scheduling, see [44]

* Corresponding author.

E-mail addresses: irfan.younas@nu.edu.pk, irfany@kth.se (I. Younas), kamrani@kth.se (F. Kamrani), maryam.bashir@nu.edu.pk (M. Bashir), schubert@foi.se (J. Schubert).