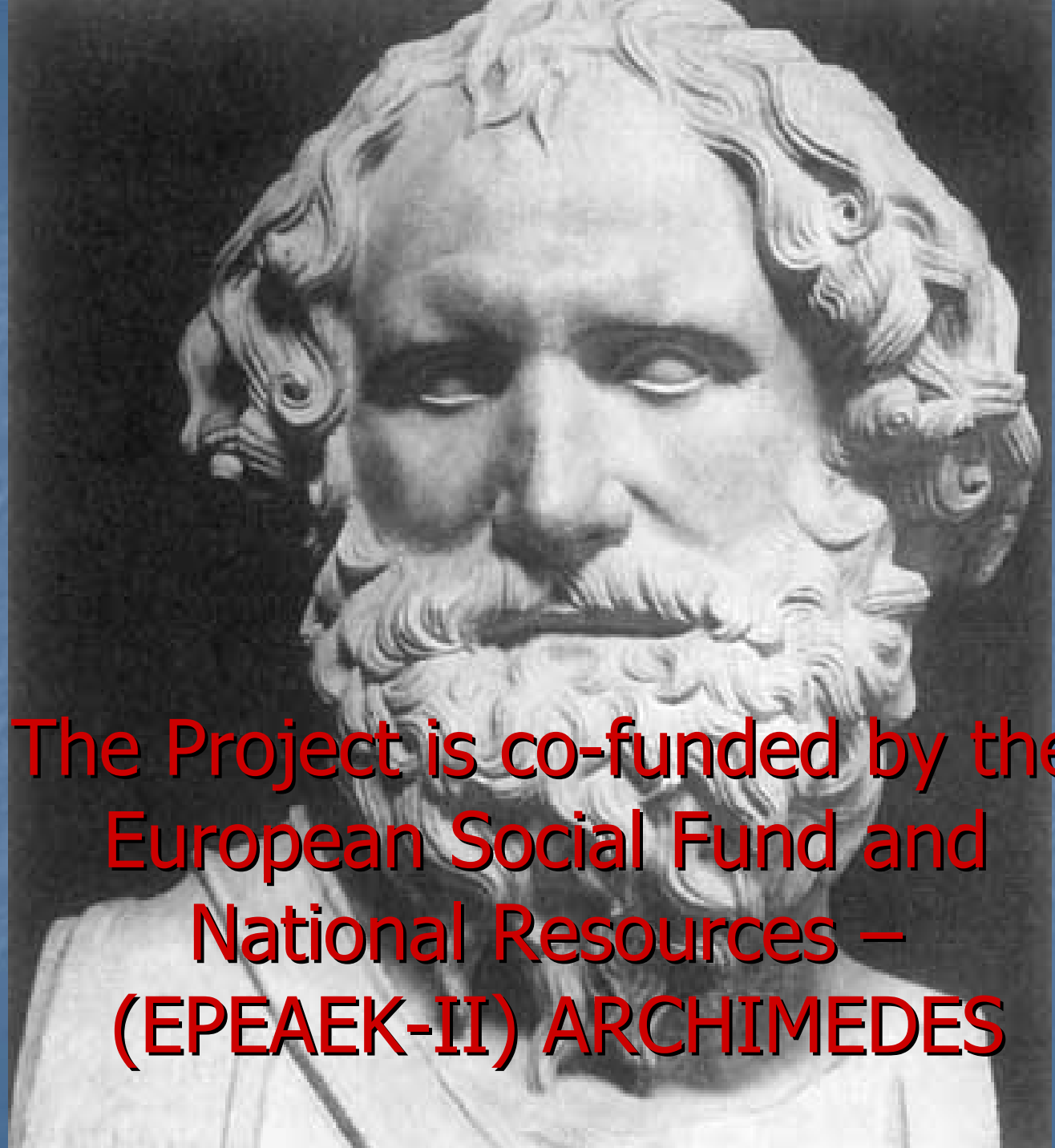
Evolutionary Timetabling with a Priority-Based Indirect Representation

1. Spyros Kazarlis,
Technological Educational Institute of Serres Greece,
Dept. of Informatics & Communications

2. Vassilios Petridis,
3. Panagiotis Adamidis,
4. Paulina Fragkou



The Project is co-funded by the European Social Fund and National Resources – (EPEAEK-II) ARCHIMEDES

Abstract

When applying evolutionary algorithms to optimization problems, the solution representation issue is of critical importance to the overall performance. This work presents an indirect representation method for genetically encoding partially described solutions of educational timetabling problems that are completed by a "timetable builder" algorithm using priorities to allocate events. A genetic algorithm using this technique is applied on several real problem instances retrieved from Greek universities, and the solutions are compared to the corresponding man-made ones, with promising results.

Timetabling Problems

- According to A. Wren : "Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives."

- Timetabling problems as well as scheduling problems, define a class of hard-to-solve constrained optimization problems of combinatorial nature (mostly NP complete)

- In this work we focus on University Timetabling Problems that split into two categories :

1. Course (lecture) Timetabling, and
2. Exam Timetabling

University Course Timetabling Problems

- Our work considers University Course Timetabling Problems, that can be described as follows :
- Find the exact time allocation within a limited time period (e.g. a week), of a number of events (courses-lectures) and also assign to them a number of resources (a teacher, a room, etc.) in such a way that a number of constraints are satisfied.

MONDAY	TUESDAY	WEDNESDAY	
08:00 – 10:00 Course : Calculus Teacher: Brown Room : A19	08:00 – 10:00 Course : CAD Teacher: Foster Room : C02	08:00 – 10:00 Course : Unix Teacher: Poter Room : A04	
10:00 – 12:00 Course : Physics Teacher: Smith Room : B23	10:00 – 12:00 Course : Java Teacher: Alister Room : D11	10:00 – 12:00 Course : Design Teacher: Harris Room : C01	. . .
. . .	. . .	. . .	

Problem Constraints

- The constraints that have to be satisfied by a timetable are usually divided into two categories: hard constraints and soft ones.
- Hard constraints are those constraints that must be rigidly fulfilled. Examples of such constraints are:
- - No resource (teacher, student, room, etc.) may be assigned to different events at the same time.
- - Events of the same semester must not be assigned at the same time slot (in order for the students of the semester to be able to attend all semester lessons).
- - Assigned resources to an event (e.g. teachers) must belong to the set of valid resources for that event (e.g. only specific teachers can teach a specific course).

Problem Constraints

- On the other hand, soft constraints are those that it is desirable to be fulfilled to the possible extent, but are not fully essential for a valid solution. Therefore, soft constraints can also be seen as optimization objectives for the search algorithm. Examples of such constraints are:
- - Schedule an event within a particular “window” of the whole period (e.g. on evenings).
- - Minimize time gaps or travel times between adjacent lectures of the same teacher.
- - Schedule multiple events of the same course at the same room

Existing Methods

- A large number of methods have been already proposed in the literature for solving timetabling problems:

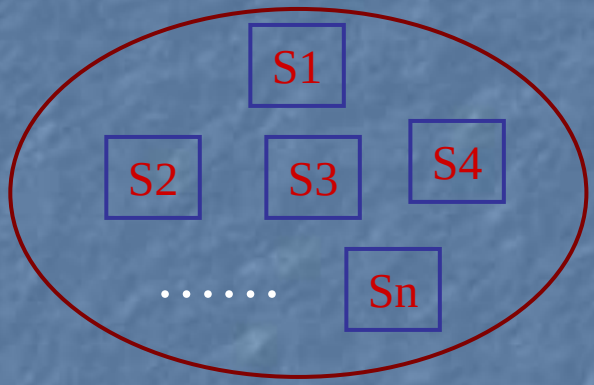
1) Sequential Methods that treat timetabling problems as graph problems.
2) Cluster Methods, in which the problem is divided into a number of event sets so that each set satisfies all hard constraints. Then, the sets are assigned to time slots.
3) Constraint Based Methods, according to which a timetabling problem is modeled as a set of variables (events) to which values (resources) have to be assigned in order to satisfy a number of constraints, and
4) Meta-heuristic methods, such as genetic algorithms, simulated annealing, tabu search, and other heuristic approaches, that are mostly inspired from nature.

Genetic Algorithms

- GAs are naturally inspired algorithms that maintain and evolve a population of DNA-like encoded solutions to the problem, using nature-like operations and techniques.
- They mimic Darwin's evolution theory, aiming at evolving good solutions to the problem, as nature evolves fit species to the environment.
- They have wide applicability and proven efficiency in solving difficult large scale optimization problems.
- Although they cannot intrinsically handle constraints, there already exists a large number of methods, proposed in the literature, for alleviating this inefficiency.
- They can be easily hybridized with other known methods to increase overall optimization performance
- They do not guarantee the discovery of the optimal solution in limited run time

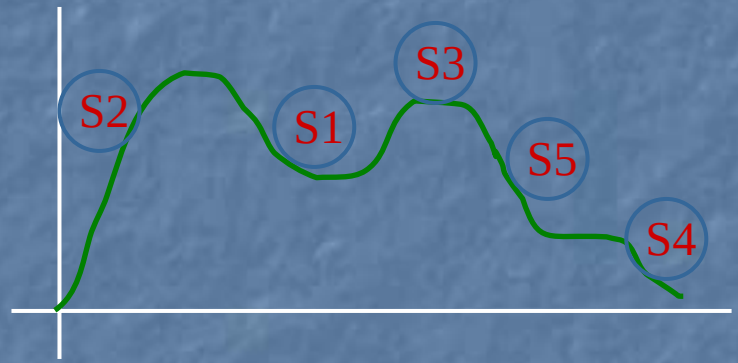
Genetic Algorithms' Principles

Population of Solutions

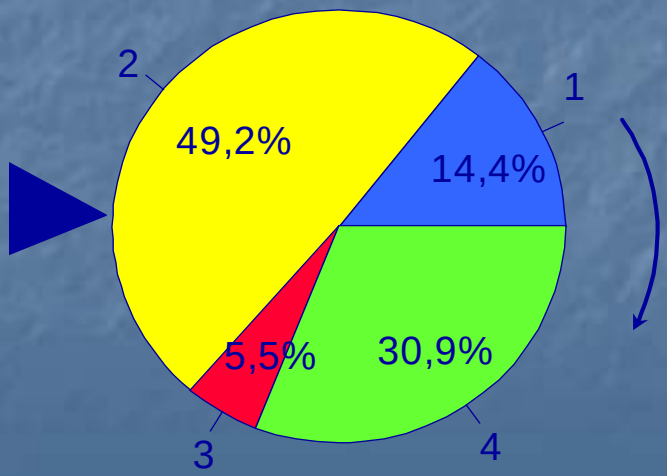


Si, i=1..n :

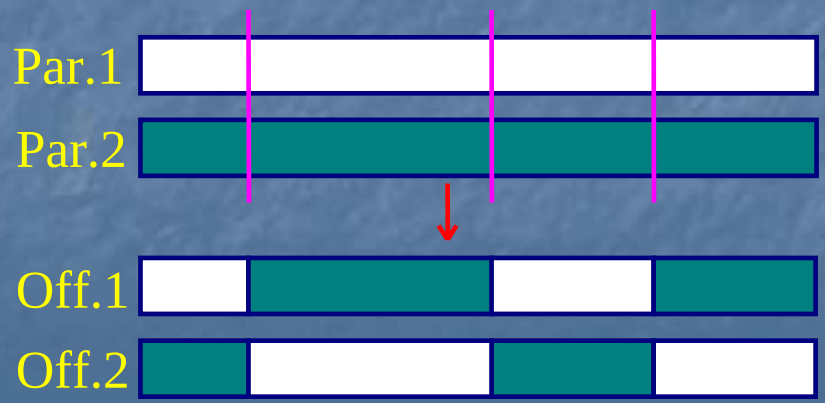
Distribution of solutions in the search space



Roulette Wheel parent selection



The Crossover Operator



Encoding of Problem Solutions

- When applying evolutionary algorithms to optimization problems, the solution representation issue is of critical importance to the overall performance.
- Two categories of encoding techniques exist :

1) Direct Representations, that directly encode the whole and exact solution with every detail, to a symbol string (e.g. binary string).

2) Indirect Representations, that encode an intermediate and partially abstract form of the solution, that has to be transformed to a full-detailed solution by some straightforward working algorithm.

Direct Representations

- Have the following characteristics:

- They are usually easily implemented
- They produce large genotypes (encoded solutions) and therefore large search spaces
- They leave the constraint satisfaction issue totally to the "hands" of the GA
- They produce very complicated search spaces, especially when constraints are considered.

Indirect Representations

- Have the following characteristics:

- They are usually more difficult to implement.
- The level of solution abstraction must be carefully chosen in order to balance between genotype simplicity and easiness of transforming to a valid solution
- They produce smaller genotypes (encoded solutions) and therefore smaller search spaces
- They need an algorithm to transform the abstract solution to an analytical one
- Many constraints can be satisfied during the "stretching" of the intermediate solution.
- They produce less complicated search spaces, as many constraints are absent from the search space.

The proposed Indirect Representation

- In our method we encode the following fields for every allocatable event :

1) The Day to allocate the event (e.g. Monday, Tuesday,..)
2) The Teachers (1..n) to assign to the event
3) The Room where the event will be held
4) A Priority figure according to which the event will be allocated within the day.

- We do not encode the exact time slot of the day, in which the event will be allocated.
- This is to be defined by the straightforward algorithm, called "Timetabler", that deploys the encoded solution to a full-detailed one
- The priority figure defines the order in which the events of the day are processed by this algorithm.

The Timetabler Algorithm

1) It separates events into clusters, one for each day.
2) For each cluster, it sorts the events according to their priority value
3) It takes the first event in the cluster, marks it as taken, and tries to place it into the schedule of the particular day.
4) Starting from time slot 1 it places the event and checks if any constraints are violated. If not the allocation is fixed.
5) If any constraints are violated, it tries to allocate the event into subsequent time periods.
6) If there exists no such time period, the event is marked to violate the "maximum time periods per day" constraint.
7) The algorithm continues with the next event in the list. When all events have been processed, the "timetabler" moves to the next cluster (day), for all days in the schedule horizon.

The Real Test Cases

- As test cases we have used the Timetable Problems of :

1) The Dept. of Electr.Eng. (Aristotle Univ. of Thessaloniki)
2) The Dept. of Informatics & Communcations (Technological Educational Institute of Serres, Greece)

Characteristic	Electr. Eng.	Inform. & Comm.
Number of Courses	79	71
Number of Lectures	104	187
Number of Events	161	192
Number of Semesters	5	7
Course Types	7	2
Number of Teachers	110	82
Number of Rooms	14	17
Number of Days	5	5
Max time-slots per day	12	13
Encoded solution length	4347 bits	3456 bits

The Hard Constraints Considered

No	Hard Constraint Description
1	No resource (teacher or room) may be assigned to different events at the same time
2	Events of the same semester must not be assigned at the same time when they belong to certain course types (e.g. when both events are of type "theory")
3	Maximum no of time periods per day cannot be exceeded
4	Each lecture may be assigned at specific rooms
5	Each room may have its own availability schedule
6	Each lecture may be assigned to specific teachers
7	Specific lectures must be rigidly assigned to specific pre-defined teachers, that usually belong to the staff.

The Hard Constraints Considered b'

No	Hard Constraint Description
8	Events of same lecture should have a minimum no-of days distance between them
9	Total weekly hours for each teacher should not exceed a maximum limit, different for each teacher.
10	Total weekly hours for each teacher should not fall below a minimum limit, different for each teacher.
11	Teacher travel times between lecture locations should not exceed the nominal interval (15 min.)
12	Student travel times between lecture locations should not exceed the nominal interval (15 min.)

The Soft Constraints Considered

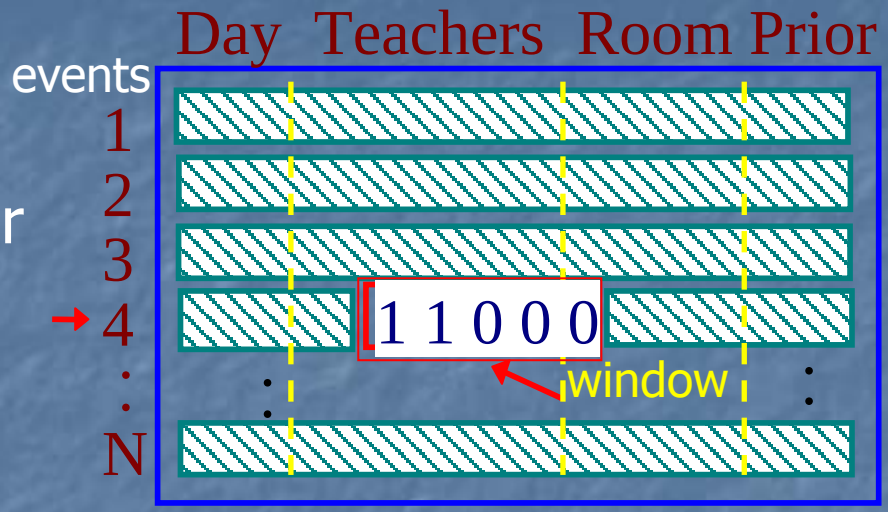
No	Soft Constraint Description
1	Every teacher has his/her own availability schedule or submits a plan with desirable time periods that suits him/her best
2	Minimize the travel time of teachers between rooms within the campus
3	Minimize the travel time of students of the same semester between rooms within the campus
4	Minimize the time gaps within the schedule of each teacher
5	Minimize the time gaps within the schedule of each room
6	Avoid allocating events at “late” hours

The GA implementation

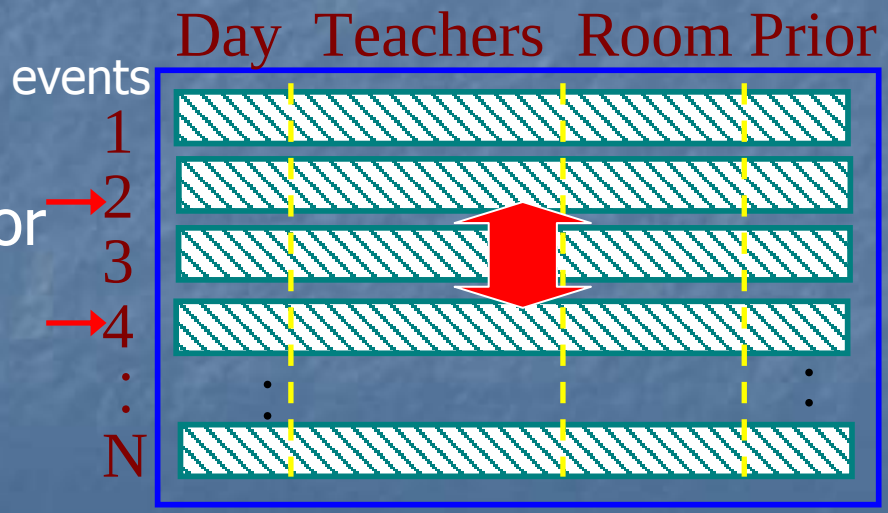
- The GA implementation used the following parameters:
- Binary encoding of the (abstract) solutions
- Population of 100 genotypes per generation
- Run limit of 5000 generations
- Roulette Wheel Parent Selection
- Multi-point binary Crossover operator
- Binary Mutation Operator
- Adaptive Operator Application Probabilities
- Elitism
- Full generational replacement of parents with offspring
- A blend of advanced combinatorial operators (Window Mutation, Swap Chromosome, Mutate Chromosome)
- A blend of general and problem-specific hill-climbing operators (mGA-comb., Fix-Day, Day-Climb)

Advanced Operators

- Windows Mutation Operator

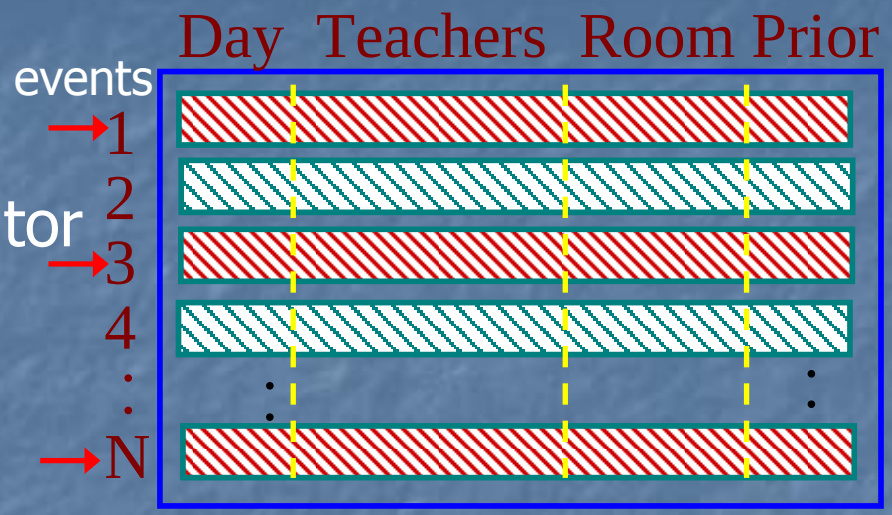


- Swap Chromosome Operator

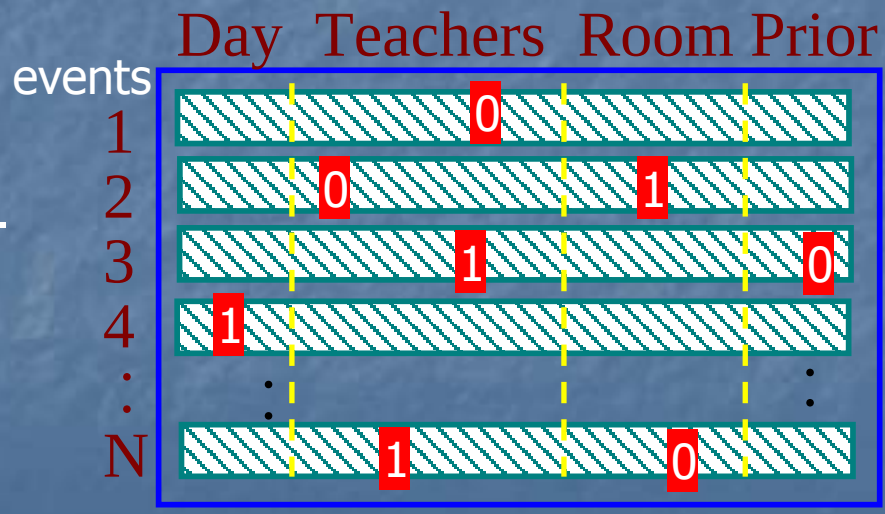


Advanced Operators b'

- Mutate Chromosome Operator



- Micro GA combinatorial hill-climbing operator



The Fitness Function

- The problems are encoded as minimization problems
- Constraints are handled using the Penalty Function Method: penalty terms are added to the objective function proportional to constraint violation
- The Fitness Function of a solution S is formulated as:

$$F(S) = O(S) + P(S) = \Sigma\, O_i(S)*k_i + \Sigma\, P_j(S)*w_j$$

- Where O(S) is the objective part (soft constraints) and P(S) is the penalty part. $O_i(S)$ are objective terms analogue to soft-constraint satisfaction, one for each objective i, and k_i are objective function coefficients (weights).
- $P_j(S)$ are penalty terms, one for each constraint j, and w_j are penalty function coefficients (weights).

Optimization Results

- A large number of GA experiments have been conducted. Here we present the best results produced, together with the corresponding man-made solutions

Χαρακτηριστικό	Electrical Engineer.	Human Solution	Informat.& Commun.	Human Solution
Overall Fitness	672	799	1104	1294
Objective Value	672	799	1104	1294
Penalty Value	0	0	0	0
No of Viol.Constraints	0	0	0	0
Room time-gaps	0	63	1	94
Teacher time-gaps	0	58	3	102

Discussion of Results

- The GA with priority-based indirect encoding finds better solutions than those produced by man.
- In the case of the Electrical Eng. dept. timetable, GA found the optimal solution, with zero constraint violations, and zero teacher and room time-gaps.
- The corresponding human solution, although feasible, it exhibits 58 and 63 time-gaps respectively.
- In the case of the Inform.& Comm. dept. timetable, GA found a solution very close to the absolute optimum, having zero constraint violations, and teacher/room time-gaps 3 and 1 respectively.
- On the other hand, the human solution has 102 and 94 time-gaps respectively.

Conclusions and Future Work

- In this work, a new method is presented for encoding solutions of timetable problems, for the application of GAs.
- This method uses an indirect representation that encodes abstract forms of the solutions, that need to be transformed to analytical ones by a specific algorithm that eventually allocates events to time-slots.
- This method has been tested on two real course timetable problems of Greek university departments.
- The GA-produced solutions satisfy all hard constraints and are optimal or near optimal concerning soft constraints and objectives. Moreover they are significantly better compared to the man-made counterparts.
- This method could be tested on more benchmark problems, and could be compared to direct encoding GA implementations for direct comparisons.