Università degli Studi di L'Aquila - Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Course catalogue

Programme of Module "Ottimizzazione Combinatoria"

- Code: F0141
- Type of course unit: Compulsory (Bachelor Degree in Computer Science curriculum General)
- Level of course unit: Undergraduate Degrees
- Semester: 2

Number of ects credits: (Bachelor Degree in Computer Science) 6 (workload 150 hours)

Teachers: Claudio Arbib (Claudio.Arbib@univaq.it)

1	**Course objectives**	Learn algorithmic techniques for some combinatorial optimization problems. Being able to formulate and solve combinatorial optimization problems using integer linear programming. Understand the complexity of the problems studied.
2	**Course content and learning outcomes (dublin descriptors)**	Topics of the module include: • Prerequisites: graphs, definitions and basic properties. Elements of computational complexity: classes P, NP; polynomial-time reduction and the class NP-complete. Examples. Combinatorial optimization problems. Basic definitions. Examples: Transversal (vertex-cover), stable set, dominating set, edge-cover, (perfect) matching, knapsack, graph colouring (chromatic number and index), shortest path, spanning subgraph etc. Formulation as 0-1 linear programming. • Unimodular and totally unimodular matrices. Necessary conditions, sufficient conditions. Hoffmann-Kruskal's Theorem. Convex hull and ideal formulation of an integer LP. Examples: shortest path, min-cost flow, bipartite matching, interval graphs. • Recursion in CO: paths on directed acyclic graphs (DAG) and dynamic programming. Examples of application: optimal paths in a DAG, 0-1 Knapsack. [Projection of polyhedra, systems of linear inequalities, Fourier-Veronese's Theorem, Fourier-Motzkin's Method. Fundamentals of duality theory in LP. The Primal-Dual Method. Example of application: Dijkstra's Algorithm]. • Matroids and the Greedy Algorithm. Definition and characterization of a matroid (Rado's Theorem). Basics of linear algebra: linear/affine dependence-independence, bases, uniqueness of representation, Steinitz's (or exchange) Theorem. Examples: trivial matroid, graphical matroid, partition matroid, vector matroid, [matching matroid]. Matroid representation. [An industrial application]. [Rank function, submodular and supermodular set functions. Rank of a matroid. Support function of a convex set. Submodular polyhedron. Lòvasz's extension. Matroid polyhedron, matroid intersection, intersection of two submodular polyhedra]. • Guaranteed approximation algorithms for CO. Definitions. Polynomial approximation schemes (PTAS, EPTAS, FPTAS). Examples: TSP: double tree and Christofides; Knapsack 0-1. • Matching (perfect/non-perfect, weighted/unweighted, bipartite/non-bipartite). Weak dual inequalities: transversal, stability number, edge-cover and matching. The bipartite case: Gallai's and König's Theorem. [Bi-stochastic matrices and perfect bipartite matching. Non-bipartite matching: matching polyhedron, Edmonds' Theorem]. • Linear relaxation of a (mixed) integer linear program. Implicit enumeration: the Branch-and-Bound Method. Linear bounds, example: integer Knapsack, 0-1 Knapsack. Dichotomy on a fractional variable. Combinatorial bounds, example: TSP. • [Separation of an integer polyhedron. Tips on the Ellipsoid Method. Separation and optimization: Finding valid inequalities. Solution of LP with exponentially many variables, example: Min-cut, TSP, 0-1 Knapsack, Stable Set]. On successful completion of this module, the student should : • Being able to evaluate the general complexity of combinatorial optimization problems. Being able to formulate them as 0-1 linear programming. Know basic primal-dual relations. Know elementary matroid theory, dynamic programming. Have the notion and know the main properties of unimodular matrices. Have basic notions on

algorithm and problem complexity. Know standard algorithms for spanning tree, bipartite matching, shortest path. Know heuristics for the TSP and Knapsack. Know general implicit enumeration, LP based algorithms (branch-and-bound)

- Understand the difference between an "easy" and a "hard" problem. Evaluate the complexity of an algorithm and, for simple cases, of a problem. Be able to recognize whether a matrix is, or is not, totally unimodular. Solve by standard algorithms the spanning tree, bipartite matching, knapsack, optimal path, travelling salesman problem. Formulate combinatorial optimization problems, or binary optimization problems derived from applications, as 0-1 linear programming.
- Understand when a problem has the shape of a matroid and what is the advantage. Know what is the advantage of a totally unimodular matrix. Decide when it is convenient to use dynamic programming. Realize the limits of heuristics.
- Prove rigorously a simple theorem. Prove or disprove (by counterexample) a simple conjecture. Understand the role played by combinatorial optimization in applications.
- Address submodular set functions, polyhedral combinatorics, advanced integer linear programming.

3	**Course prerequisites**	Notion on basic algorithms and linear algebra, linear programming
4	**Teaching methodsand language**	lectures, exercises **Language**: Italian **Reference textbooks** • W.J. Cook, W. H. Cunningham, W. R. Pulleyblank, A. Schrijver, ***Combinatorial Optimization***. Wiley. 1997. • C.H. Papadimitriou, K. Steiglitz, ***Combinatorial Optimization: Algorithms and Complexity***. Dover Books on Computer Science. 1998. *http://www.amazon.com/Combinatorial-Optimization-Algorithms-Complexity-Computer/dp/0486402584* • A. Sassano,, ***Modelli e Algoritmi della Ricerca Operativa***. Franco Angeli Editore.
5	**Assessment methods**	written test, oral test