8. Conclusions

Roberto Amadini

Department of Computer Science and Engineering, University of Bologna, Italy

Combinatorial Decision Making and Optimization

2nd cycle degree programme in Artificial Intelligence

University of Bologna, Academic Year 2024/25

Combinatorial problems

- Solving a combinatorial problem means finding a combination of feasible values for the variables of the problem
  - Variables $\equiv$ decisions
  - Domains $\equiv$ possible choices for each decision
  - Constraints $\equiv$ restrictions defining feasible decisions
- It is often (NP-)hard to solve combinatorial problems
  - "Easy" to verify if a solution is feasible
  - "Hard" to actually find a solution
- Often we have to address combinatorial optimization
  - Objective function $\equiv$ goal (minimize cost / maximize profit)

Combinatorial problems

- How to face these problems?
- Exact algorithms: they guarantee to find an optimal solution although this may take exponential time
  - The focus of this course
- Approximation algorithms: they guarantee in polynomial time a (sub-)optimal solution at most ρ times worse the optimal one
  - $\rho =$ approximation factor
- Heuristic algorithms: no guarantee of optimality nor polynomial runtime, but "in practice" they find good solutions in reasonable time
  - According to empirical evidences

Can a LLM solve NP-hard problems?

✅ Summary

Task	Can LLM do it?	How?
Solve small NP problems	✅	Internally or via generated code
Solve large NP problems	⚠️ Partially	With heuristics or solvers
Prove NP-completeness	✅	By reduction or explanation
Find optimal solutions efficiently	❌	Not guaranteed unless problem is small
Write solving code	✅	Very effective at this

Combinatorial problems and AI

- So, how can AI help to find exact solutions of combinatorial problems? By leveraging constraint solvers
- We need first to define a model for the problem
  - Model $\equiv$ mathematical abstraction of a real-world problem
  - Model + input data = problem instance
  - Here LLMs can help
- Then, encode the problem in your favorite paradigm and ask for solution(s) to corresponding solver(s)
  - CP, SAT, SMT, MIP, ...
  - Different technology = different solving approach
- In this module we focused on SMT and MIP solving

Satisfiability Modulo Theory

- SMT extends SAT to solve formulas in (quantifier-free) FOL over different theories
  - functions, (constants), predicates with arity > 1
- Similar/orthogonal to CP, it tackles combinatorial problems from a "more logical" perspective (formulas)
  - More oriented to problems derived from software analysis
- Eventually, SMT solving eagerly or lazily relies on SAT solving
  - Eager approach: translates upfront a SMT formula to equisatisfiable SAT formula (a.k.a. "bit-blasting")
  - Lazy approach: combine SAT solvers + $\mathcal{T}$-solvers (a.k.a. *CDCL*($\mathcal{T}$))
    - $\mathcal{T}$-information used lazily over Boolean abstractions
    - Everyone (SAT and SMT solvers) does what it is good at
    - Modular, flexible, typically more efficient than eager approach

Satisfiability Modulo Theory

- SMT solver = collection of theory solvers
- Different theory solvers have been developed for different theories
  - E.g. EUF, DL, LRA, LIA, ...
  - We often need to combine theories (e.g., Nelson-Oppen)
  - For some of them, optimization support (OMT)
- SMT-LIB initiative started in 2003 to:
  - Provide rigorous descriptions of SMT theories
  - Develop and promote common languages for SMT solvers.
  - Connect developers, researchers and users of the SMT community
  - Establish and make available benchmarks for SMT solvers.
  - Collect and promote software tools useful to the SMT community

Mixed Integer Programming

- Linear programming (LP) is about solving problems with linear constraints/objective function in some canonical or standard form
  - LP is one of the main fields of Operation Research
  - The feasible region for a LP problem is a convex polyhedron
- The simplex algorithm is a well-known method to tackle LP problems
  - Worst-case exponential, typically polynomial
  - Worst-case polynomial algorithms exist for LP problems (interior point)
- We can solve the dual of a LP problem
  - Different perspective, same optimal value
  - Dual simplex, sensitivity analysis, Benders' decomposition

Mixed Integer Programming

- Adding integer variables to LP (IP, MIP) significantly increases its complexity
  - Rounding non-integral solutions of linear relaxation does not work
- Branch-and-bound: divide-et-impera approach, branches on variables with non-integer value, stop if we cannot improve incumbent solution
- Cutting planes: linear equalities separating non-integral, optimal solutions of linear relaxation from feasible region of original problem
  - Branch-and-cut, Gomory's cut, Benders' cut
- Nonlinear programming: use ad hoc techniques or linearize
  - Big-M, Unary encoding

Which technology?

Technology	Best used when...
CP	- Constraints are global and have no “smooth” structure over finite-domain variables - The objective function is not a large sum of terms
SAT/SMT	- Problem involves Boolean variables and logical formulas - SMT can handle rich data types (arrays, trees, ...)
MIP	- Most constraints are linear, possibly on continuous variables - The objective is a large sum of terms (e.g., cost functions)

Which domain?

Technology	**Typical domains**
CP	Scheduling, timetabling, puzzles, resource allocation
SAT	Logic circuit checking, model checking, formal verification, planning
SMT	Software verification, theorem proving, (dynamic) symbolic execution
MIP	Supply chain optimization, finance, transportation, network flow

What we haven't seen

- Non-integer CP variables: set of integers, strings, reals, ...
  - P.J. Stuckey, G. Tack, M. Garcia de la Banda: "Twelve Years of MiniZinc" `https://freuder.files.wordpress.com/2019/09/2007.pdf`
- Max-SAT
- Pseudo-Boolean optimization (PBO)
- Quantified Boolean Formula (QBF)
- Constraint Logic Programming (CLP)
- Answer-Set Programming (ASP)
- Quadratic Programming (QP)
- ...
- Algorithm Selection and portfolio solvers

Exam

- Common project for the 2 modules
- Group project (2–4 people)
- Project proposals are welcome!
    - Subject to approval
- Discussions can also include individual questions over course topics
- Dates
    - July 2025: Submission 06/07, discussions week of July 14
    - September 2025·: Submission 06/09, discussions week of September 15
- Project description and template on virtuale platform