SOLVING NP-HARD OPTIMIZATION PROBLEMS: A COMPARISON OF CLASSICAL APPROXIMATION ALGORITHMS AND MODERN METAHEURISTIC AND LEARNING-BASED METHODS

Nihat Rustamov
Faculty of SABAH, Baku State University, Baku, Azerbaijan
https://doi.org/10.5281/zenodo.21450092

ARTICLE INFO

Received: 14th July 2026
Accepted: 16th July 2026
Online: 18th July 2026

KEYWORDS

NP-hard problems; approximation algorithms; approximation ratio; metaheuristics; simulated annealing; genetic algorithms; combinatorial optimization; machine learning for optimization

ABSTRACT

Many optimization problems of practical importance in computer science are NP-hard, meaning that no polynomial-time algorithm is known that solves them exactly [1,2]. Because exact solution is computationally infeasible for large instances, the discipline has developed two broad families of practical alternatives. Classical approximation algorithms sacrifice optimality in exchange for polynomial running time while providing a provable guarantee on the quality of the solution returned [2,3]. Modern metaheuristics, including simulated annealing, genetic algorithms and ant colony optimization, abandon formal guarantees in favour of flexible stochastic search that frequently produces excellent solutions in practice [4,5]. More recently, learning-based methods have been proposed which train neural models to construct or improve solutions directly from problem instances [6]. This thesis compares these paradigms with respect to solution quality, theoretical guarantees, computational cost and generality. It argues that approximation algorithms remain essential where worst-case assurance is required, that metaheuristics dominate where empirical performance on large instances is the governing criterion, and that a hybrid strategy—using approximation results as a principled starting point for metaheuristic refinement—offers the most robust approach for practical deployment [3,5,7]..

1 Introduction

Combinatorial optimization problems arise throughout computer science and its applications, in domains as varied as network routing, scheduling, resource allocation, circuit design and logistics [1,2]. A substantial proportion of these problems belong to the class of NP-hard problems, for which no polynomial-time exact algorithm is known and, under the widely held assumption that P differs from NP, none exists [1]. Since exhaustive search becomes intractable as instance size grows, practitioners must accept solutions that are good rather than provably optimal.

Two distinct traditions have emerged in response. The first, rooted in theoretical computer science, develops approximation algorithms: polynomial-time procedures

accompanied by a mathematical proof bounding how far the returned solution can deviate from the optimum [2,3]. The second, arising largely from applied and heuristic research, develops metaheuristics: general-purpose stochastic search frameworks that explore the solution space through mechanisms often inspired by physical or biological processes, and that offer no formal guarantee but frequently perform very well empirically [4,5].

A third and more recent direction applies machine learning to combinatorial optimization, training models to construct solutions or to guide search [6]. This thesis examines all three approaches, compares their theoretical and practical characteristics, and considers how they may be combined. The objective is to establish which paradigm is appropriate under which operational conditions, rather than to identify a single universally superior method.

2 Theoretical Foundations

The complexity class NP contains those decision problems whose candidate solutions can be verified in polynomial time. A problem is NP-hard if every problem in NP reduces to it in polynomial time, and NP-complete if it is additionally a member of NP [1]. The practical consequence is that a polynomial-time exact algorithm for any NP-hard problem would yield one for all problems in NP, an outcome regarded as improbable.

Approximation theory formalizes the notion of a near-optimal solution. An algorithm is said to be a ρ-approximation for a minimization problem if, for every instance, the cost of the solution it returns is at most ρ times the optimal cost [2,3]. The value ρ, termed the approximation ratio, is a worst-case guarantee: it holds for every input, not merely for typical ones. Problems differ sharply in how well they can be approximated. Some admit a polynomial-time approximation scheme, permitting a ratio arbitrarily close to one at the cost of increased running time. Others are provably inapproximable beyond a fixed threshold unless P equals NP, a result established through the theory of probabilistically checkable proofs [3].

3 Classical Approximation Algorithms

Classical approximation algorithms are constructed so that their output quality can be bounded analytically. Several design paradigms recur throughout the literature [2,3].

3.1 Greedy and Combinatorial Techniques

Greedy algorithms build a solution incrementally, at each step making the choice that appears locally best according to a fixed criterion. Despite their simplicity, greedy methods yield strong guarantees for several important problems. The greedy algorithm for set cover, for instance, achieves a logarithmic approximation ratio, and this is essentially the best possible unless P equals NP [2,3]. The appeal of such algorithms lies in their transparency and low computational cost, which make them suitable as components of larger systems.

3.2 Linear Programming Relaxation and Rounding

A widely used technique formulates the problem as an integer linear program, relaxes the integrality constraints to obtain a linear program solvable in polynomial time, and then rounds the fractional solution to an integral one [3]. The optimal value of the relaxation provides a bound against which the rounded solution can be compared, yielding the approximation guarantee. Randomized rounding, in which fractional values are interpreted as

probabilities, has proved particularly powerful and underlies approximation results for covering, packing and facility location problems [3].

3.3 Strengths and Limitations

The decisive advantage of approximation algorithms is the guarantee itself. In safety-critical or contractual settings, an assurance that the solution is within a known factor of optimal is qualitatively different from an empirical observation that a method usually performs well [2,3]. Approximation algorithms are also typically fast and deterministic, and their analysis illuminates the structure of the problem.

Their limitation is that the guarantee is a worst-case bound, and worst-case bounds can be pessimistic. An algorithm with a factor-two guarantee may in practice return solutions within a few percent of optimal, but it may equally return solutions near the bound. Moreover, for many problems the best achievable ratio is provably large, so the guarantee, while rigorous, may be too weak to be useful. Finally, approximation algorithms are problem-specific: each requires bespoke design and analysis, and a technique developed for one problem rarely transfers directly to another [3,4].

4 Metaheuristic and Learning-Based Methods

Metaheuristics address precisely the weaknesses of problem-specific design. They provide general search frameworks applicable, with modest adaptation, across a wide range of problems [4,5].

4.1 Local Search and Simulated Annealing

Local search improves an incumbent solution by repeatedly moving to a superior neighbour, terminating at a local optimum. Simulated annealing extends this by occasionally accepting inferior moves with a probability that decreases over time according to a cooling schedule, in analogy with the annealing of metals [4]. This mechanism permits escape from local optima, and under idealized conditions the method converges to a global optimum, although the required schedule is impractically slow.

4.2 Population-Based Methods

Genetic algorithms maintain a population of candidate solutions and evolve it through selection, crossover and mutation, exploiting the recombination of partial solutions [5]. Ant colony optimization constructs solutions probabilistically using pheromone trails that reinforce components appearing in good solutions. Such population-based methods explore the search space broadly and have been applied successfully to routing, scheduling and design problems, frequently outperforming approximation algorithms in solution quality on realistic instances [4,5].

4.3 Learning-Based Approaches

Recent research trains neural models to solve combinatorial problems directly, either by constructing solutions sequentially or by learning heuristics that guide conventional search [6]. Reinforcement learning has been used to train policies that build tours for routing problems, and graph neural networks exploit the relational structure inherent in many combinatorial instances. The attraction of this approach is that a model trained on a distribution of instances may amortize its cost across many subsequent problems. The limitations are substantial: performance degrades on instances unlike those seen in training,

the training process is computationally expensive, and no guarantee accompanies the output [6,7].

5 Comparative Evaluation

The paradigms differ systematically across the criteria that matter in practice [3,5,6].

The most fundamental distinction concerns theoretical assurance. Classical approximation algorithms provide a provable worst-case bound on the quality of every solution they return, whereas metaheuristics and learning-based methods provide no such guarantee and rest instead on empirical evidence of good performance [2,3,4]. This difference is not merely technical: an assurance that holds for all inputs is qualitatively distinct from an observation that a method has performed well on the inputs tried so far.

Empirical solution quality frequently inverts this ordering. Because approximation algorithms are designed to satisfy a worst-case bound, they are often conservative, and the solutions they produce, while provably acceptable, are rarely optimized aggressively for the instance at hand. Metaheuristics, by contrast, search the solution space adaptively and commonly attain solutions far closer to optimal on realistic instances than any approximation bound would suggest [4,5].

Generality is a further point of divergence. Approximation algorithms are problem-specific: each requires bespoke design and a tailored proof, and a technique developed for one problem seldom transfers directly to another. Metaheuristics offer general frameworks applicable across a wide range of problems with only modest adaptation, which substantially reduces the engineering effort required to address a new optimization task [4,5].

The paradigms also differ in their computational profile and reproducibility. Approximation algorithms run in predictable polynomial time and are typically deterministic, so repeated execution on the same input yields the same output. Metaheuristics are stochastic, producing different results across runs, and their running time is tunable rather than fixed, permitting the practitioner to trade additional computation for improved solution quality. This flexibility comes with a corresponding burden: metaheuristic performance depends heavily on parameter tuning, whereas approximation algorithms expose few parameters requiring adjustment [4,5,7].

The pattern is consistent: approximation algorithms trade empirical quality for rigour and predictability, while metaheuristics trade rigour for empirical quality and flexibility. Learning-based methods extend the metaheuristic trade-off further, offering potential amortization of cost across many instances at the price of generalization risk when the instances encountered depart from the training distribution [6].

6 Discussion

The comparison suggests that the choice of method should be governed by the requirements of the application rather than by a general claim of superiority. Where a worst-case assurance is contractually or operationally necessary—in resource allocation subject to regulatory scrutiny, for example, or in systems where an unbounded failure is unacceptable—an approximation algorithm is the appropriate instrument, and its conservatism is the price of certainty [2,3].

Where the governing criterion is instead the quality of the solution actually obtained on the instances encountered in practice, and where extensive empirical validation is feasible, metaheuristics are generally preferable. Their capacity to exploit additional computation time is particularly valuable, since a practitioner can trade running time for quality according to the operational budget [4,5].

These observations point toward hybridization rather than exclusive choice. An approximation algorithm can supply a high-quality initial solution together with a proven bound on its cost; a metaheuristic can then refine that solution, using the bound to quantify the remaining optimality gap. The resulting procedure inherits the guarantee of the former and the empirical performance of the latter, and the approximation bound provides a principled stopping criterion that purely heuristic search lacks [3,5,7].

7 Conclusion

NP-hard optimization problems admit no efficient exact solution, and practical computation must therefore proceed by approximation of one kind or another. This thesis has compared classical approximation algorithms, which provide provable worst-case guarantees, with metaheuristic and learning-based methods, which provide flexibility and strong empirical performance without formal assurance [2,4,6].

Neither family is uniformly superior. Approximation algorithms are indispensable where rigour is required; metaheuristics are preferable where empirical quality on realistic instances governs; learning-based methods are promising but remain constrained by generalization and cost. The most robust practical strategy is hybrid: an approximation algorithm establishes a guaranteed baseline, and metaheuristic refinement improves upon it while the theoretical bound quantifies the remaining gap [3,5,7]. Future work should focus on strengthening the theoretical understanding of learning-based methods so that they may eventually offer guarantees comparable to those of classical approximation.

References:

1. Garey M R, Johnson D S. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: W. H. Freeman; 1979.
2. Vazirani V V. Approximation Algorithms. Berlin: Springer; 2003.
3. Williamson D P, Shmoys D B. The Design of Approximation Algorithms. Cambridge: Cambridge University Press; 2011.
4. Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by Simulated Annealing. Science. 1983;220(4598):671–680.
5. Holland J H. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press; 1975.
6. Bengio Y, Lodi A, Prouvost A. Machine Learning for Combinatorial Optimization: A Methodological Tour d'Horizon. European Journal of Operational Research. 2021;290(2):405–421.
7. Cormen T H, Leiserson C E, Rivest R L, Stein C. Introduction to Algorithms. 4th ed. Cambridge: MIT Press; 2022.