



WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Featured content](#)
[Current events](#)
[Random article](#)
[Donate to Wikipedia](#)
[Wikipedia store](#)

Interaction
[Help](#)
[About Wikipedia](#)
[Community portal](#)
[Recent changes](#)
[Contact page](#)

Tools
[What links here](#)
[Related changes](#)
[Upload file](#)
[Special pages](#)
[Permanent link](#)
[Page information](#)
[Wikidata item](#)
[Cite this page](#)

Print/export
[Create a book](#)
[Download as PDF](#)
[Printable version](#)

Languages
[Deutsch](#)
[فارسی](#)
[Français](#)
[한국어](#)
[Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

Branch and cut

From Wikipedia, the free encyclopedia

Branch and cut^[1] (sometimes written as *branch-and-cut*) is a method of [combinatorial optimization](#) for solving [integer linear programs](#) (ILPs), that is, [linear programming](#) (LP) problems where some or all the unknowns are restricted to [integer](#) values.^[2] Branch and cut involves running a [branch and bound](#) algorithm and using [cutting planes](#) to tighten the linear programming relaxations. Note that if cuts are only used to tighten the initial LP relaxation, the algorithm is called **cut and branch**.

Contents [hide]

- [1 Description of the Algorithm](#)
- [2 Branching Strategies](#)
- [3 External links](#)
- [4 References](#)

Description of the Algorithm [\[edit\]](#)

This description assumes the ILP is a maximization problem.

The method solves the [linear program without the integer constraint](#) using the regular [simplex algorithm](#). When an optimal solution is obtained, and this solution has a non-integer value for a variable that is supposed to be integer, a cutting plane algorithm may be used to find further linear constraints which are satisfied by all feasible integer points but violated by the current fractional solution. These inequalities may be added to the linear program, such that resolving it will yield a different solution which is hopefully "less fractional".

At this point, the [branch and bound](#) part of the algorithm is started. The problem is split into multiple (usually two) versions. The new linear programs are then solved using the simplex method and the process repeats. During the branch and bound process, non-integral solutions to LP relaxations serve as upper bounds and integral solutions serve as lower bounds. A node can be pruned if an upper bound is lower than an existing lower bound. Further, when solving the LP relaxations, additional cutting planes may be generated, which may be either *global cuts*, i.e., valid for all feasible integer solutions, or *local cuts*, meaning that they are satisfied by all solutions fulfilling the side constraints from the currently considered branch and bound subtree.

The algorithm is summarized below.

1. Add the initial ILP to L , the list of active problems
2. Set $x^* = \text{null}$ and $v^* = -\infty$
3. while the L is not empty
 1. Select and remove a problem from L
 2. Solve the LP relaxation of the problem.
 3. If the solution is infeasible, go back to 3 (while). Otherwise denote the solution by x with objective value v .
 4. If $v \leq v^*$, go back to 3.
 5. If x is integer, set $v^* \leftarrow v, x^* \leftarrow x$ and go back to 3.
 6. If desired, search for cutting planes that are violated by x . If any are found, add them to the LP relaxation and return to 3.2.
 7. Branch to partition the problem into new problems with restricted feasible regions. Add these problem to L and go back to 3
4. return x^*

Branching Strategies [\[edit\]](#)

An important step in the branch and cut algorithm is the branching step. At this step, there are a variety of branching heuristics that can be used. The branching strategies described below all involve what is called **branching on a variable**.^[3] Branching on a variable involves choosing a variable, x_i , with a fractional value, x'_i , in the optimal solution to the current LP relaxation and then adding the constraints $x_i \leq \lfloor x'_i \rfloor$ and

$x_i \geq \lceil x'_i \rceil$

- **Most Infeasible Branching** This branching strategy chooses the variable with the fractional part closest to 0.5.
- **Pseudo Cost Branching** The basic idea of this strategy is to keep track for each variable x_i the change in the objective function when this variable was previously chosen as the variable to branch on. The strategy then chooses the variable that is predicted to have the most change on the objective function based on past changes when it was chosen as the branching variable. Note that pseudo cost branching is initially uninformative in the search since few variables have been branched on.
- **Strong Branching** Strong branching involves testing which of the candidate variable gives the best improvement to the objective function before actually branching on them. **Full strong branching** tests all candidate variables and can be computationally expensive. The computational cost can be reduced by only considering a subset of the candidate variables and not solving each of the corresponding LP relaxations to completion.

There are also a large number of variations of these branching strategies, such as using strong branching early on when pseudo cost branching is relatively uninformative and then switching to pseudo cost branching later when there is enough branching history for pseudo cost to be informative.

External links [\[edit\]](#)

- [Mixed Integer Programming](#)
- [BranchAndCut.org FAQ](#)
- [SCIP](#) an open source framework for branch-cut-and-price and a mixed integer programming solver
- [ABACUS - A Branch-And-Cut System](#) - open source software
- [COIN-OR Cbc](#) - open source software

References [\[edit\]](#)

1.

[^] Padberg, M. and Rinaldi, G. (1991). "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems". *Siam Review*. 60–100. doi:10.1137/1033004

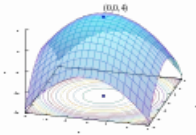
2.

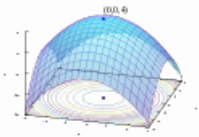
[^] John E., Mitchell (2002). "Branch-and-Cut Algorithms for Combinatorial Optimization Problems". *Handbook of Applied Optimization*. 65–77.

3.

[^] Achterberg, T.; T. Koch; A. Martin (2005). "Branching rules revisited". *Operations Research* **33** (1): 42–54. doi:10.1016/j.orl.2004.04.002

v · t · e		Optimization: Algorithms, methods, and heuristics	[hide]
		Unconstrained nonlinear: Methods calling ...	[show]
		Constrained nonlinear	[show]
		Convex optimization	[show]
		Combinatorial	[hide]
Paradigms	Approximation algorithm · Dynamic programming · Greedy algorithm · Integer programming (Branch & bound or cut)		
Graph algorithms	Minimum spanning tree	Bellman–Ford · Borůvka · Dijkstra · Floyd–Warshall · Johnson · Kruskal	
Network flows	Dinic · Edmonds–Karp · Ford–Fulkerson · Push–relabel maximum flow		
		Metaheuristics	[show]
Categories (Algorithms and methods · Heuristics) · Software			





Categories: [Combinatorial optimization](#) | [Optimization algorithms and methods](#)