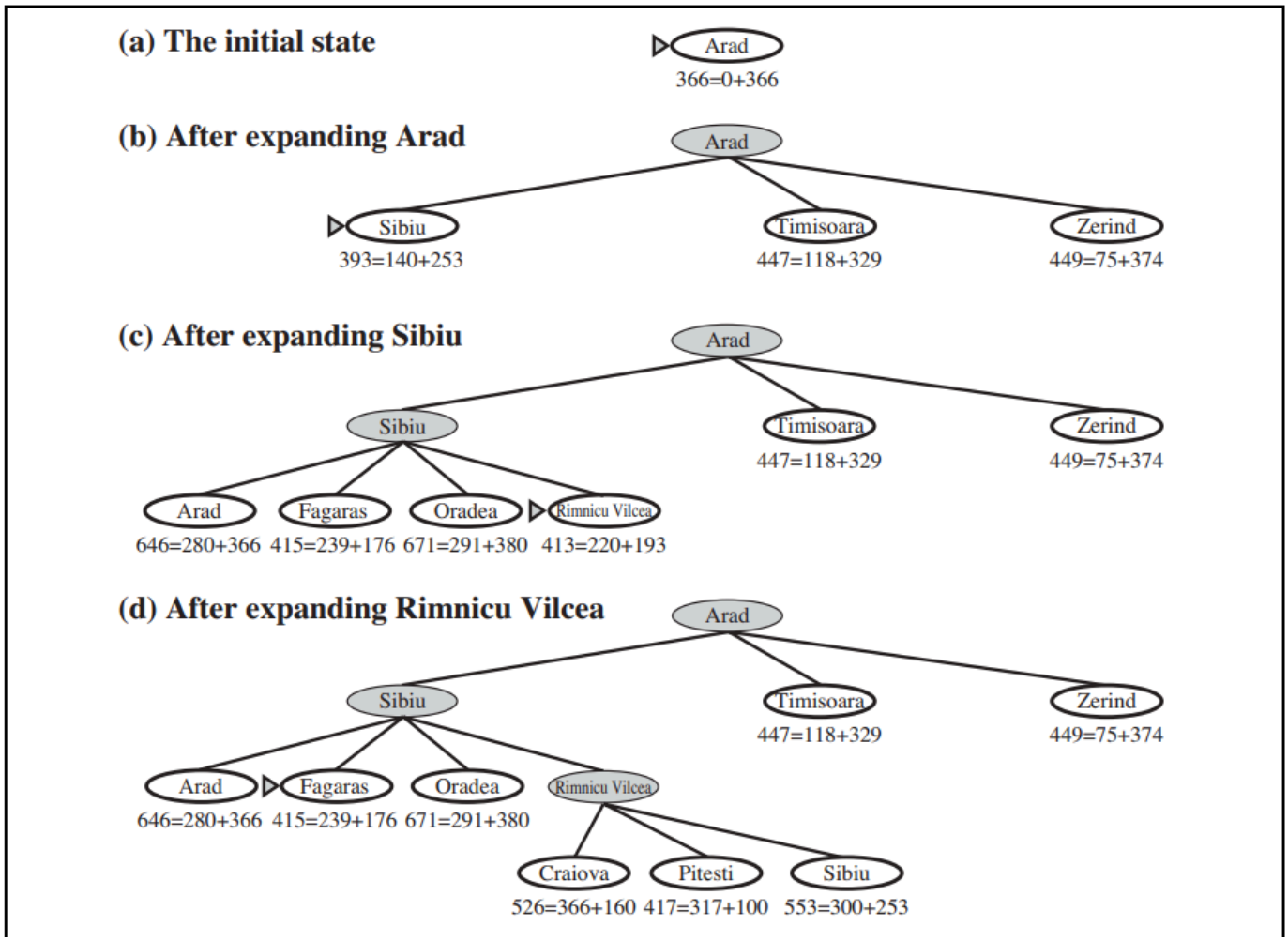
Chapter -3
Informed (HEURISTIC) Search Strategies

Informed (Heuristic) Search Techniques: Heuristic Function, Best-first Search, Greedy best-first search, Generate-and-Test, Local Search Algorithm- Hill-climbing search, Simulated annealing, And-OR Search, A* search: Minimizing the total estimated solution cost, Problem Reduction.
Constraint Satisfaction problem: Map Coloring, cryptarithmetic problem.
Adversarial Search: Games, Optimal Decision in Games, Alpha-Beta Pruning Minimax Search Procedure, Adding Alpha-beta Cut-offs, Iterative Deepening.

Best First Search -

A* Search- f(n)=g(n)+h(n)



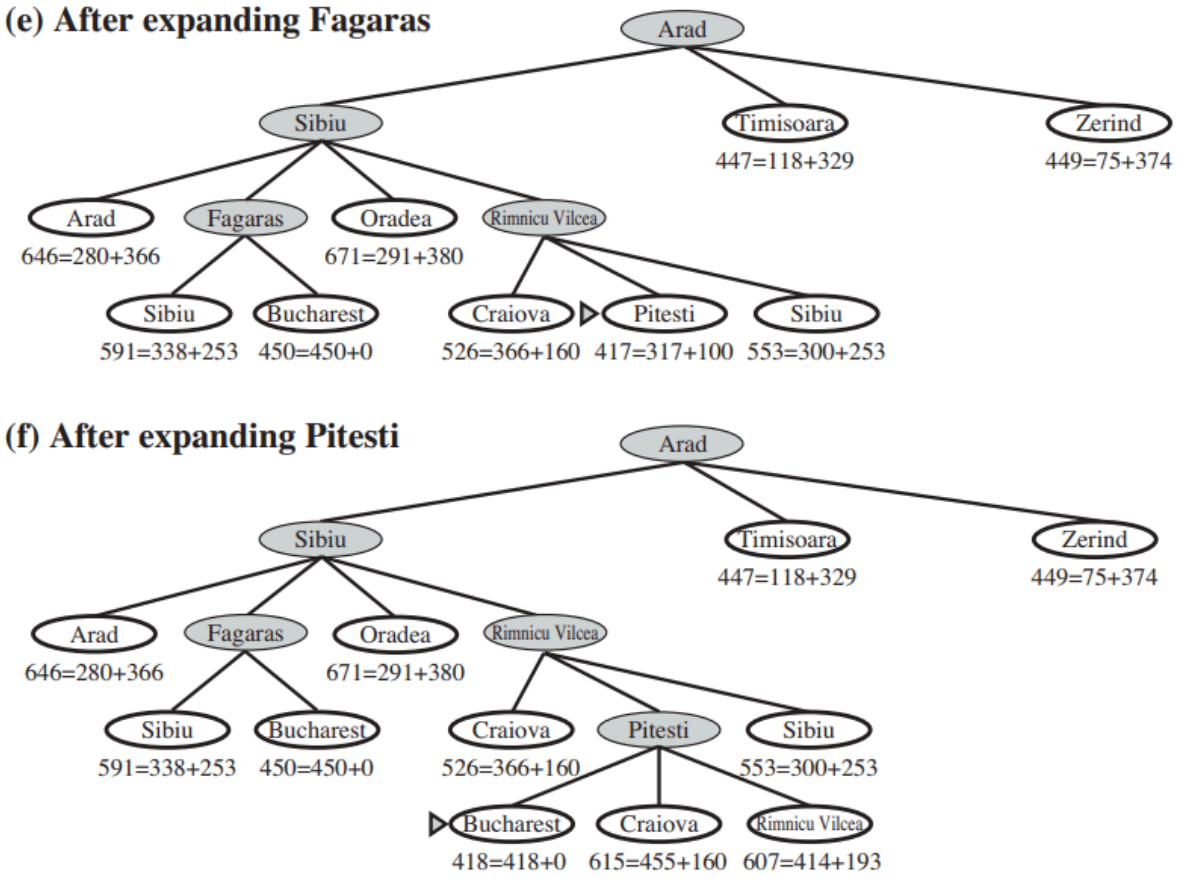


Figure 3.24 Stages in an A* search for Bucharest. Nodes are labeled with $f = g + h$. The h values are the straight-line distances to Bucharest taken from Figure 3.22.

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Figure 3.22 Values of h_{SLD}—straight-line distances to Bucharest.

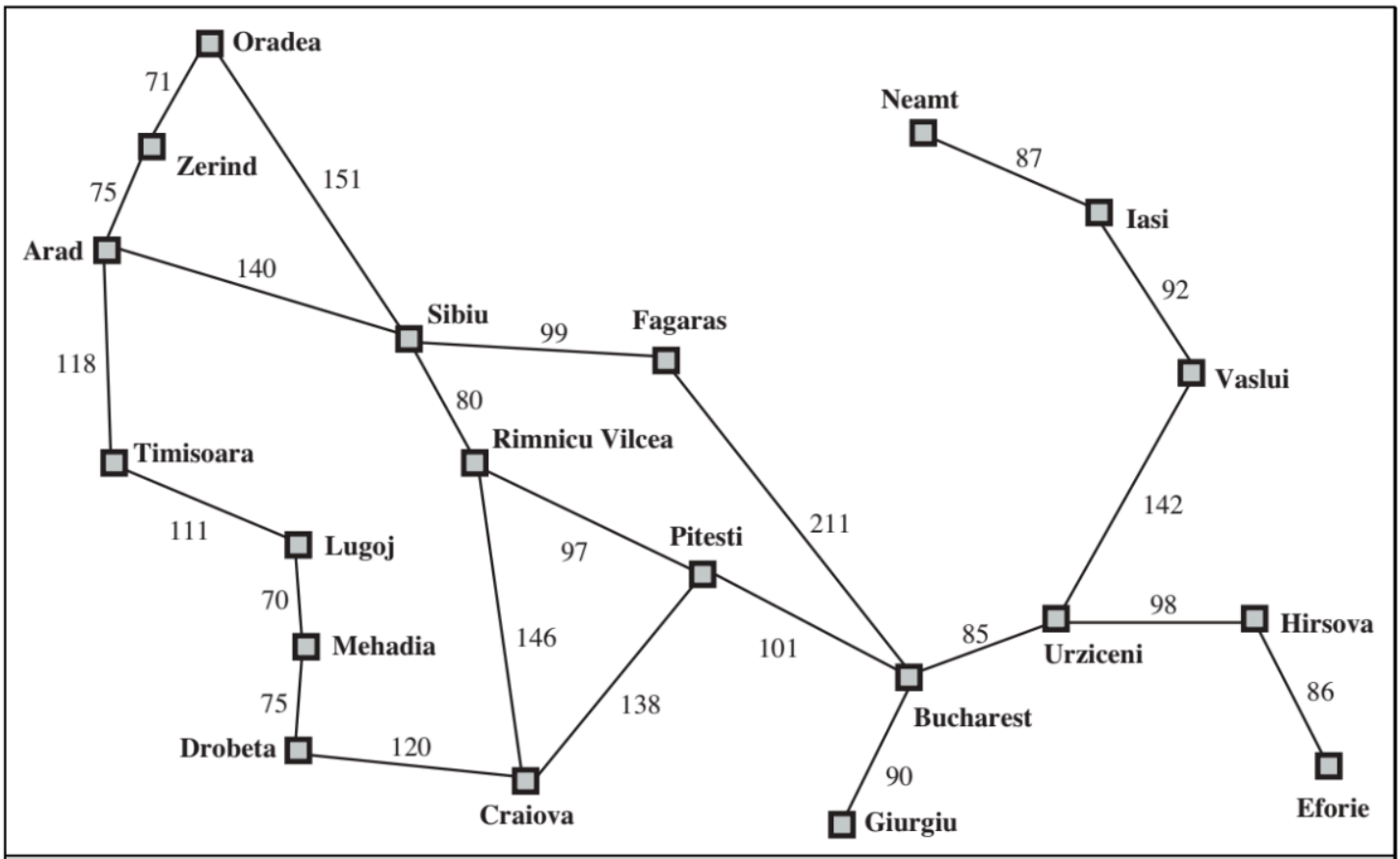


Figure 3.2 A simplified road map of part of Romania.

Heuristic function-

In this section, we look at heuristics for the 8-puzzle, in order to shed light on the nature of heuristics in general

```
function HILL-CLIMBING(problem) returns a state that is a local maximum

    current ← MAKE-NODE(problem.INITIAL-STATE)
    loop do
        neighbor ← a highest-valued successor of current
        if neighbor.VALUE ≤ current.VALUE then return current.STATE
        current ← neighbor
```

Figure 4.2 The hill-climbing search algorithm, which is the most basic local search technique. At each step the current node is replaced by the best neighbor; in this version, that means the neighbor with the highest VALUE, but if a heuristic cost estimate h is used, we would find the neighbor with the lowest h.

Local Search

- Hill Climbing

- Steepest hill Climbing

Simulated Annealing

Simulated annealing (SA) is a **probabilistic** technique for approximating the global optimum of a given function.
Specifically, it is a metaheuristic to approximate global optimization in a large search space for an optimization problem.
For large numbers of local optima, SA can find the global optima.

It is often used when the search space is discrete (for example the traveling salesman problem, the boolean satisfiability problem, protein structure prediction, and job-shop scheduling).
For problems where finding an approximate global optimum is more important than finding a precise local optimum in a fixed amount of time, simulated annealing may be preferable to exact algorithms such as gradient descent or branch and bound.

The name of the algorithm comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to alter its physical properties.

Overview-

The state *s* of some physical systems, and the function *E*(*s*) to be minimized, is analogous to the internal energy of the system in that state. The goal is to bring the system, from an arbitrary *initial state*, to a state with the minimum possible energy.

The basic iteration-

At each step, the simulated annealing heuristic considers some neighboring state *s** of the current state *s*, and probabilistically decides between moving the system to state *s** or staying in-state *s*. These probabilities ultimately lead the system to move to states of lower energy. Typically this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted.

Acceptance probabilities

Acceptance probabilities [edit]

The probability of making the transition from the current state s to a candidate new state s_{new} is specified by an *acceptance probability function* $P(e, e_{\text{new}}, T)$, that depends on the energies $e = E(s)$ and $e_{\text{new}} = E(s_{\text{new}})$ of the two states, and on a global time-varying parameter T called the *temperature*. States with a smaller energy are better than those with a greater energy. The probability function P must be positive even when e_{new} is greater than e. This feature prevents the method from becoming stuck at a local minimum that is worse than the global one.

When T tends to zero, the probability $P(e, e_{\text{new}}, T)$ must tend to zero if $e_{\text{new}} > e$ and to a positive value otherwise. For sufficiently small values of T, the system will then increasingly favor moves that go "downhill" (i.e., to lower energy values), and avoid those that go "uphill." With $T = 0$ the procedure reduces to the greedy algorithm, which makes only the downhill transitions.

In the original description of simulated annealing, the probability $P(e, e_{\text{new}}, T)$ was equal to 1 when $e_{\text{new}} < e$—i.e., the procedure always moved downhill when it found a way to do so, irrespective of the temperature. Many descriptions and implementations of simulated annealing still take this condition as part of the method's definition. However, this condition is not essential for the method to work.

The P function is usually chosen so that the probability of accepting a move decreases when the difference $e_{\text{new}} - e$ increases—that is, small uphill moves are more likely than large ones. However, this requirement is not strictly necessary, provided that the above requirements are met.

The annealing schedule

The name and inspiration of the algorithm demand an interesting feature related to the temperature variation to be embedded in the operational characteristics of the algorithm. This necessitates a gradual reduction of the temperature as the simulation proceeds. The algorithm starts initially with T

set to a high value (or infinity), and then it is decreased at each step following some *annealing schedule*—which may be specified by the user, but must end with T=0

towards the end of the allotted time budget. In this way, the system is expected to wander initially towards a broad region of the search space containing good solutions, ignoring small features of the energy function; then drift towards low-energy regions that become narrower and narrower, and finally move downhill according to the steepest descent heuristic.