Discrete Optimization

Local Search: Part VII

Goal of the Lecture

- Local search
  - more systematic presentation
    - beyond neighborhood
  - heuristics versus meta-heuristics
  - heuristics

Local Search

- States
  - either solutions or configurations

Local Search

- States
  - either solutions or configurations
- Moving from state s to one of its neighbors
  - N(s): neighborhood of s

Local Search

- States
  - either solutions or configurations
- Moving from state s to one of its neighbors
  - N(s): neighborhood of s
- Some neighbors are legal; others are not
  - L(N(s),s): set of legal neighbors

Local Search

- States
  - either solutions or configurations
- Moving from state s to one of its neighbors
  - N(s): neighborhood of s
- Some neighbors are legal; others are not
  - L(N(s),s): set of legal neighbors
- Select one of the legal neighbors
  - S(L(N(s),s),s); selection function

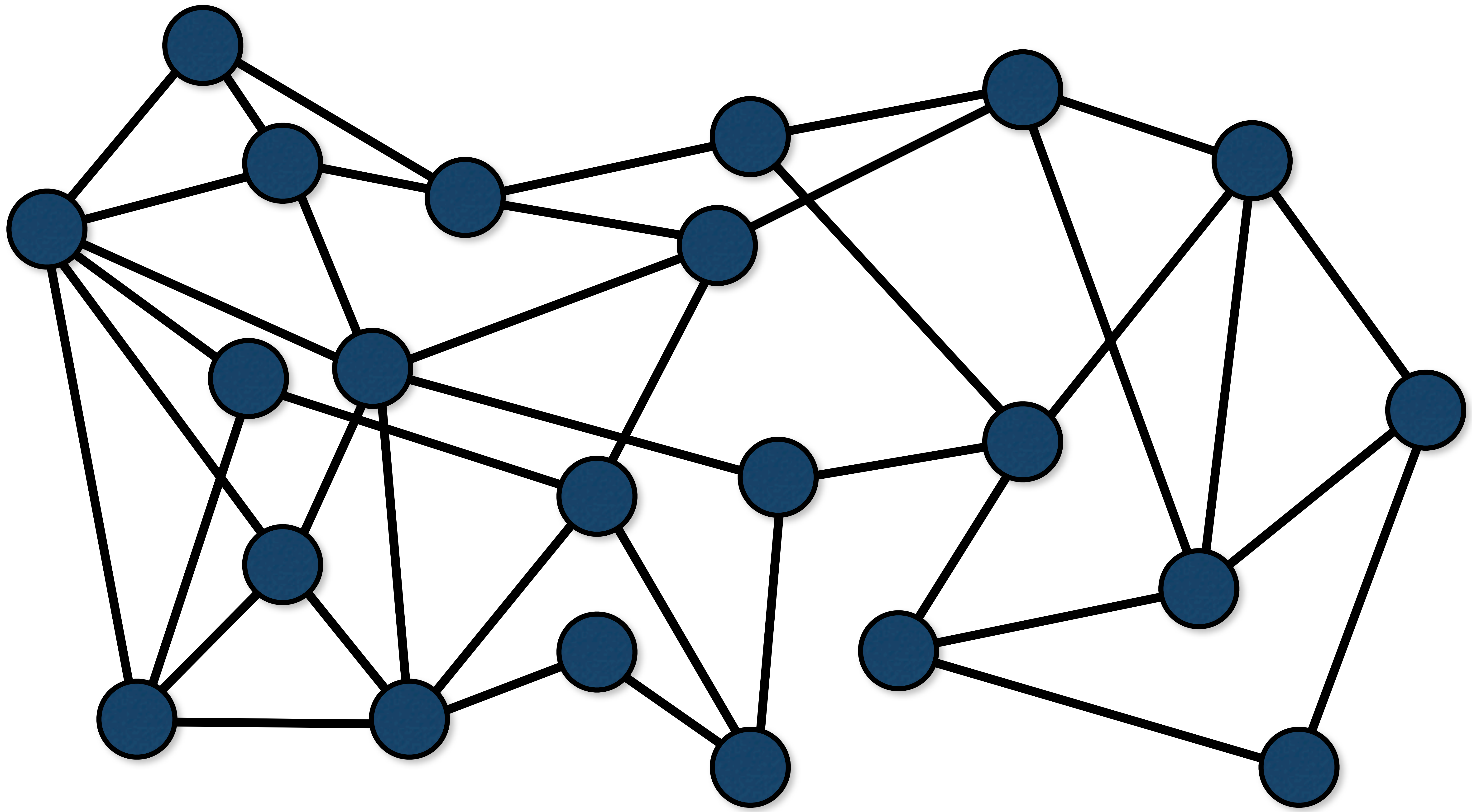
Local Search

- States
  - either solutions or configurations
- Moving from state s to one of its neighbors
  - N(s): neighborhood of s
- Some neighbors are legal; others are not
  - L(N(s),s): set of legal neighbors
- Select one of the legal neighbors
  - S(L(N(s),s),s); selection function
- Objective function
  - minimizing f(s)

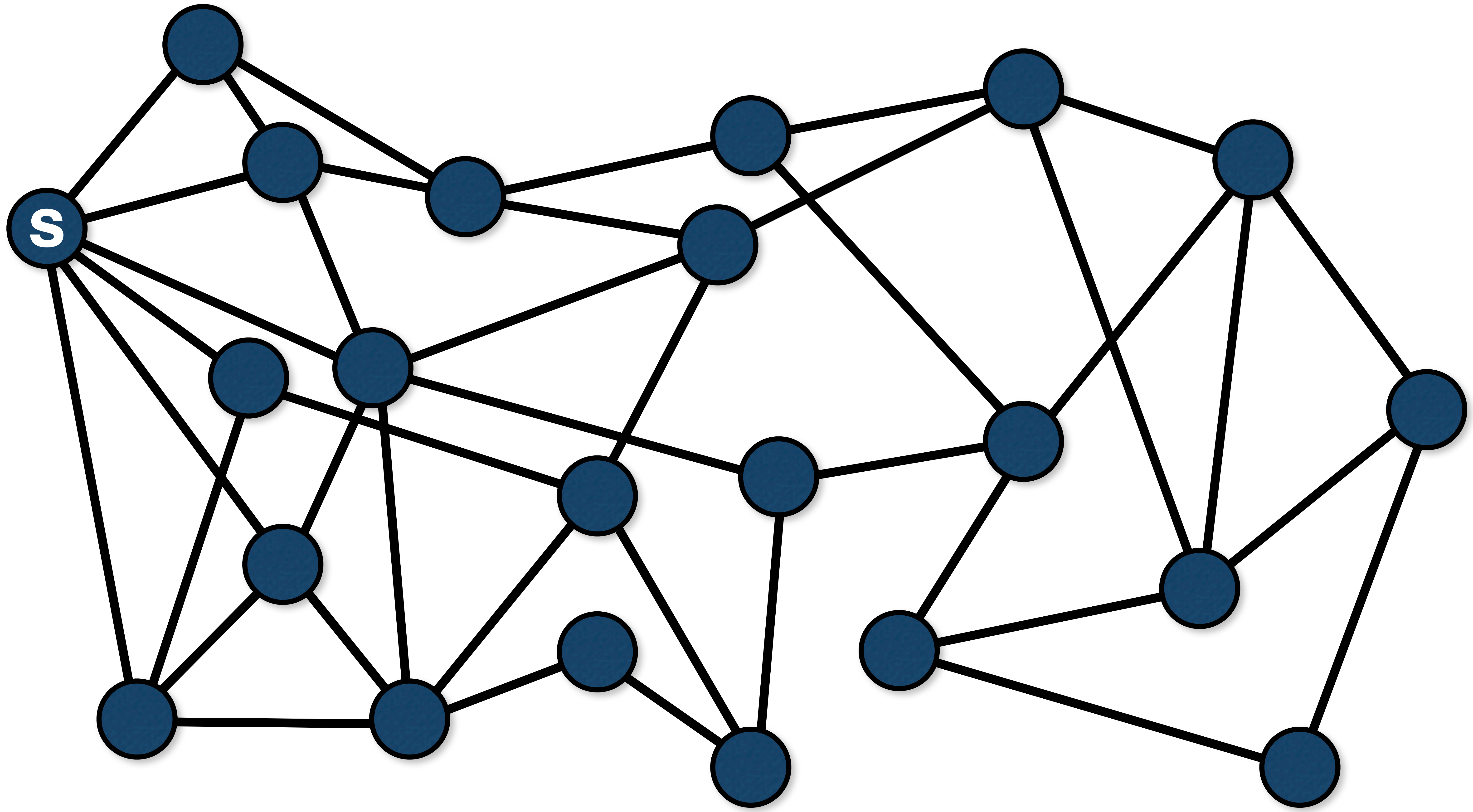
The Basic Local Search

```
function LocalSearch($f, N, L, S$) {
    $s :=$ generateInitialSolution();
    $s^* := s$;
    for $k := 1$ to $MaxTrials$ do
        if $satisfiable(s) \wedge f(s) < f(s^*)$ then
            $s^* := s$;
        $s := S(L(N(s), s), s)$;
    return $s^*$;
}
```

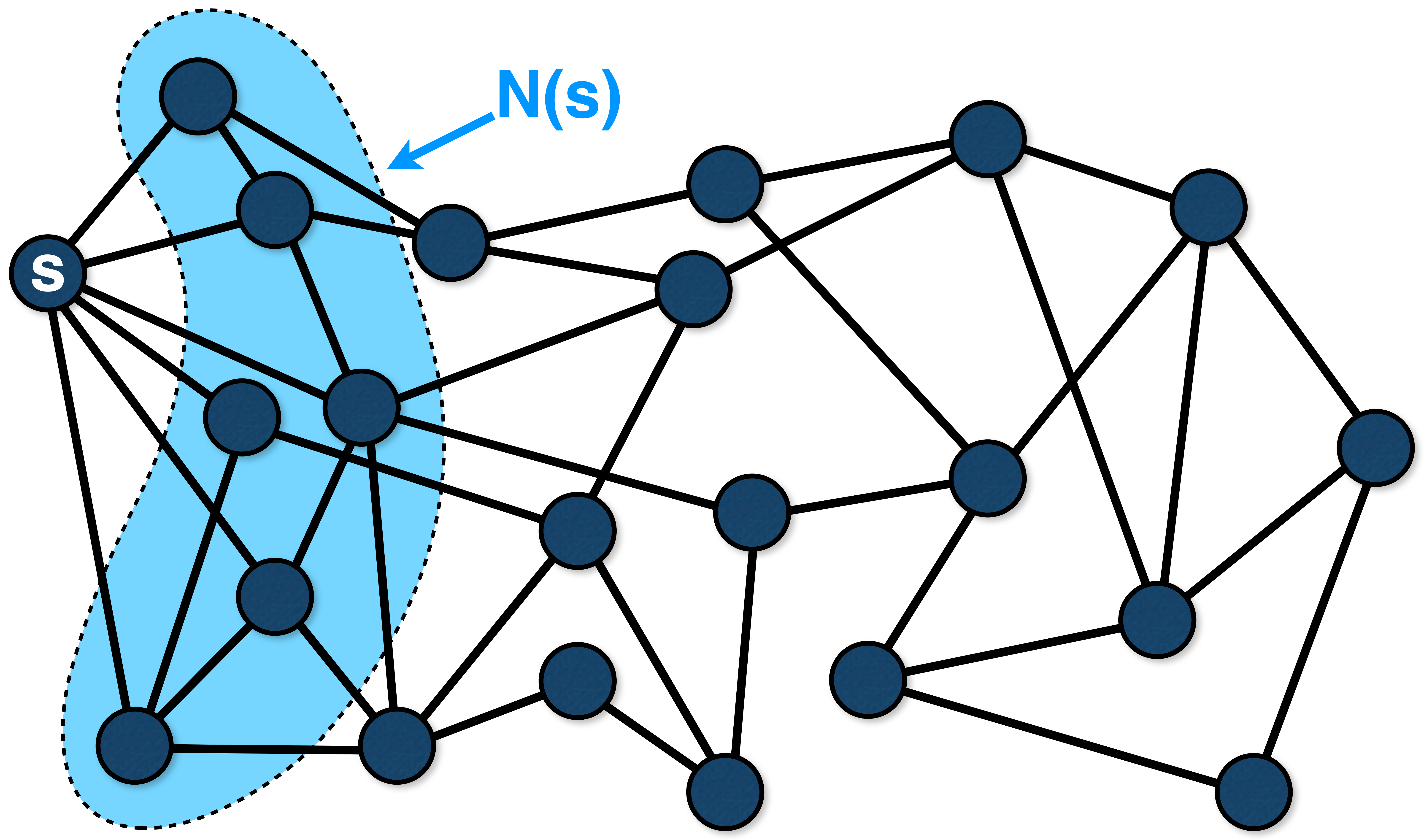
The Basic Local Search



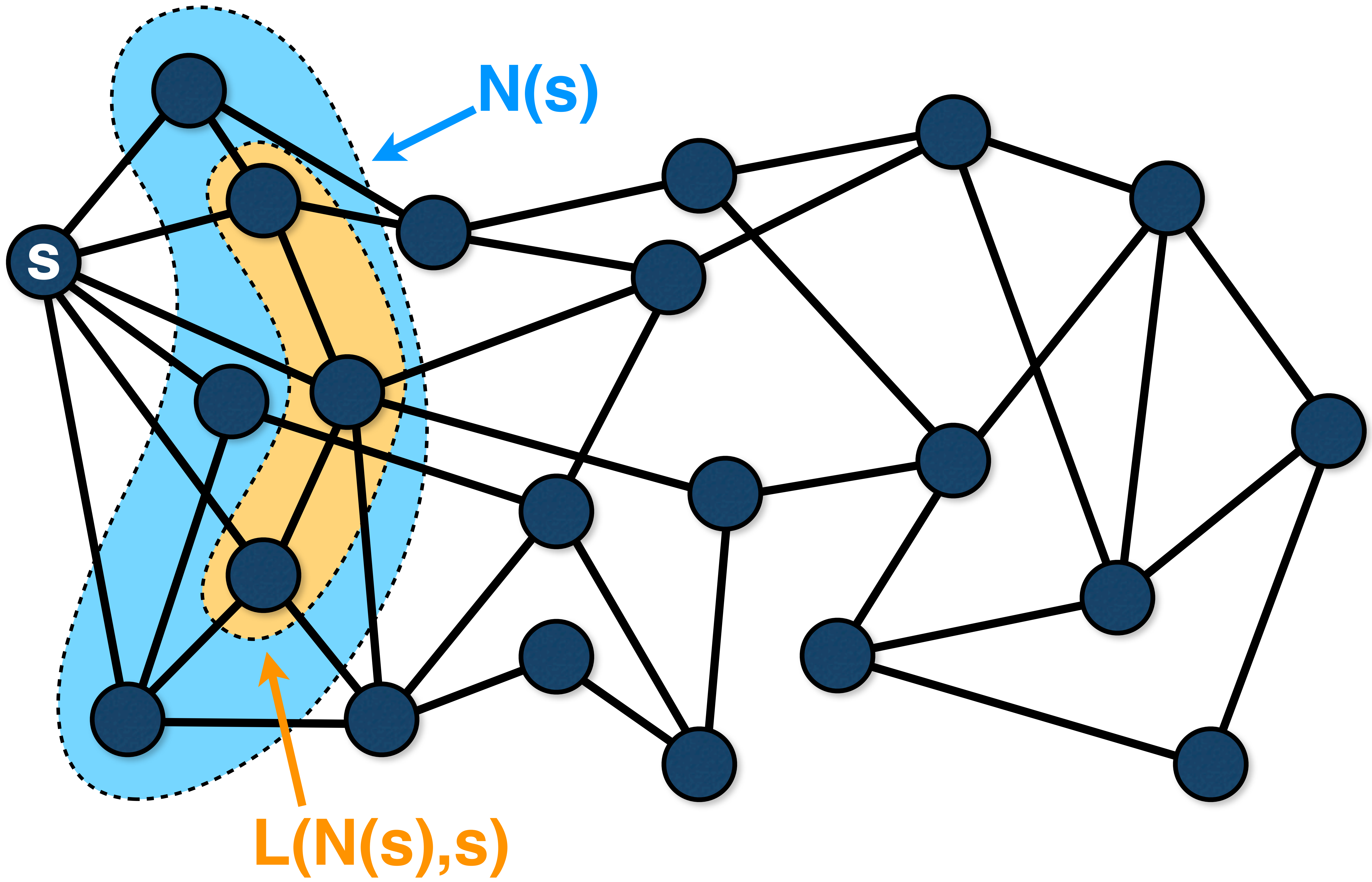
The Basic Local Search



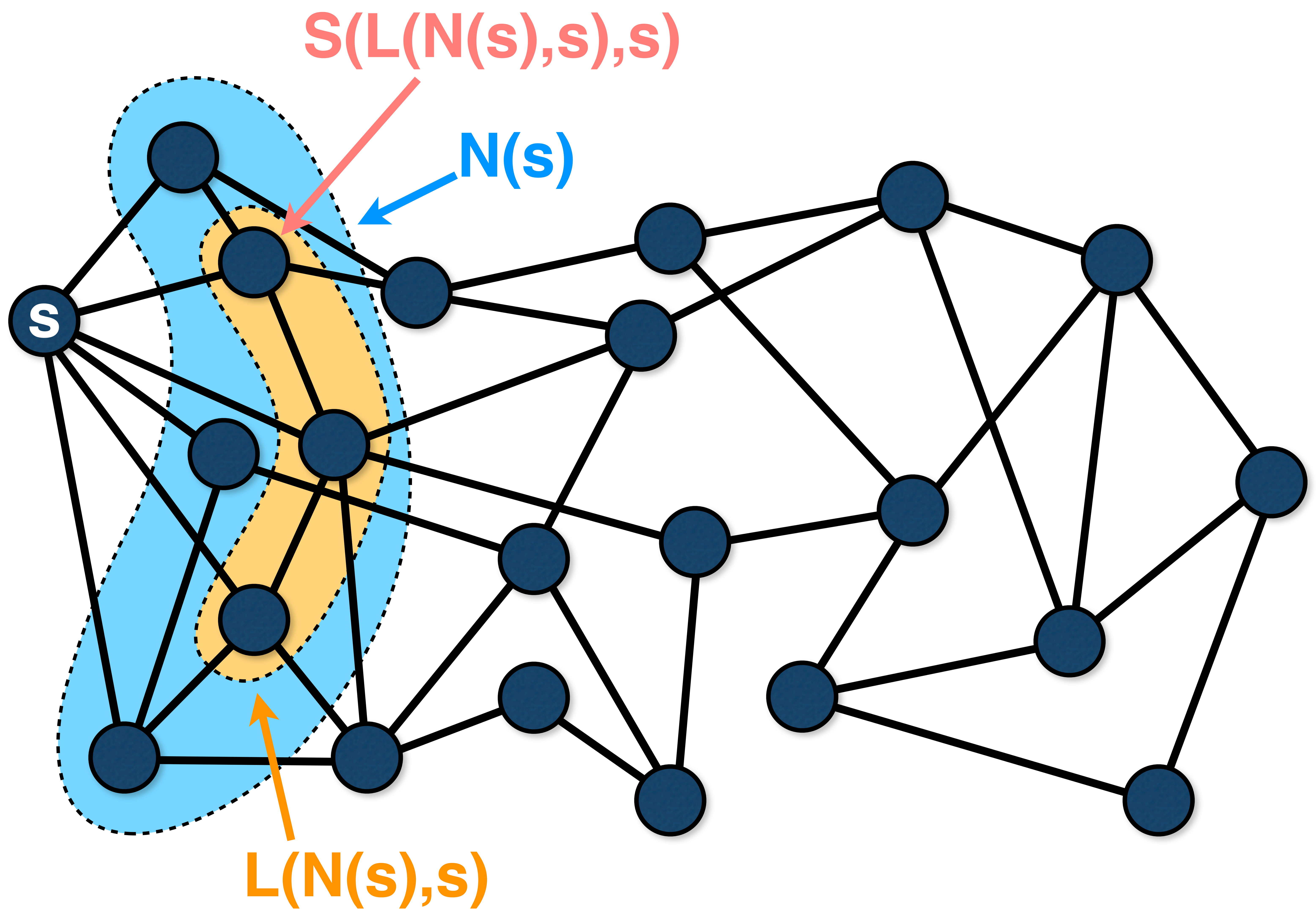
The Basic Local Search



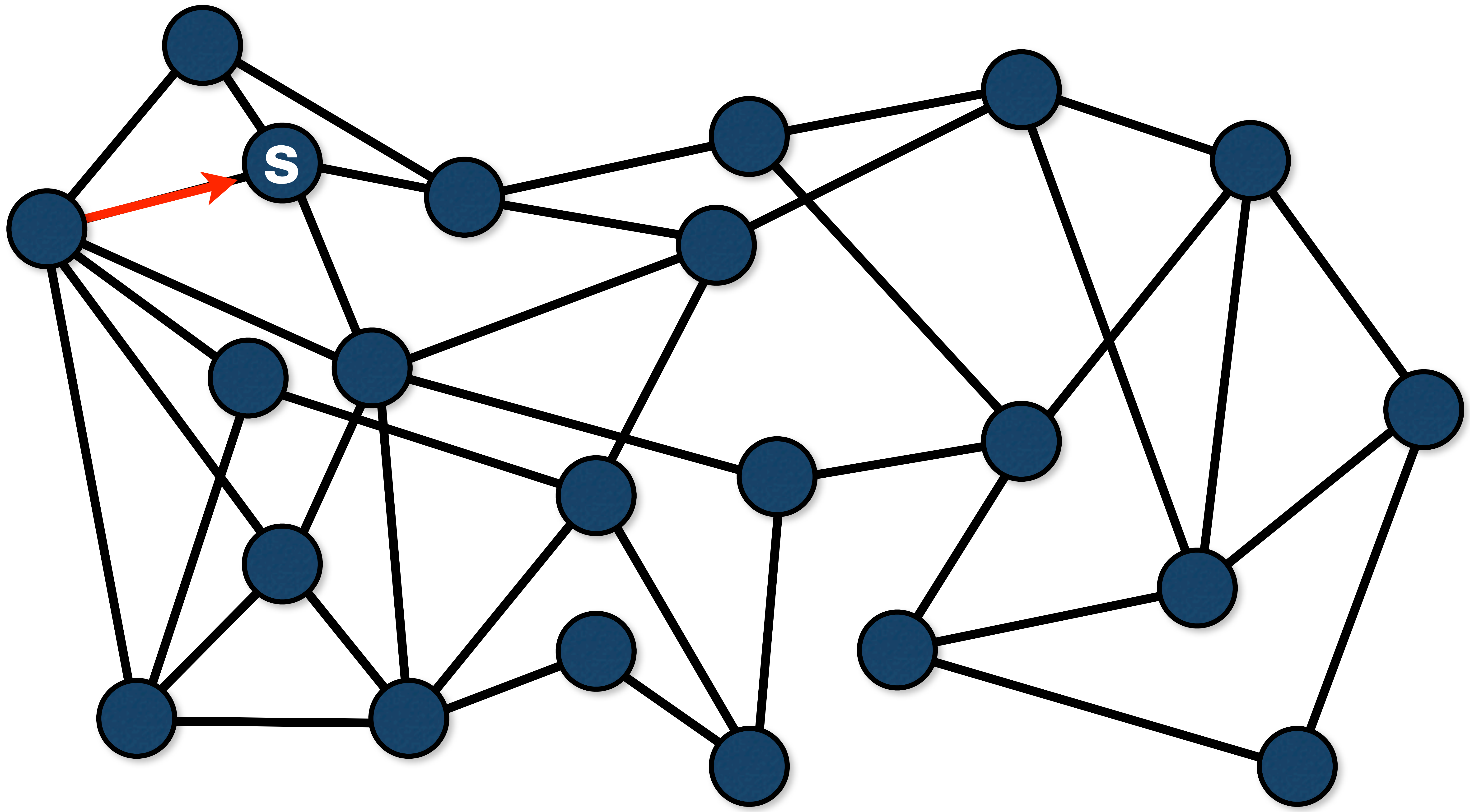
The Basic Local Search



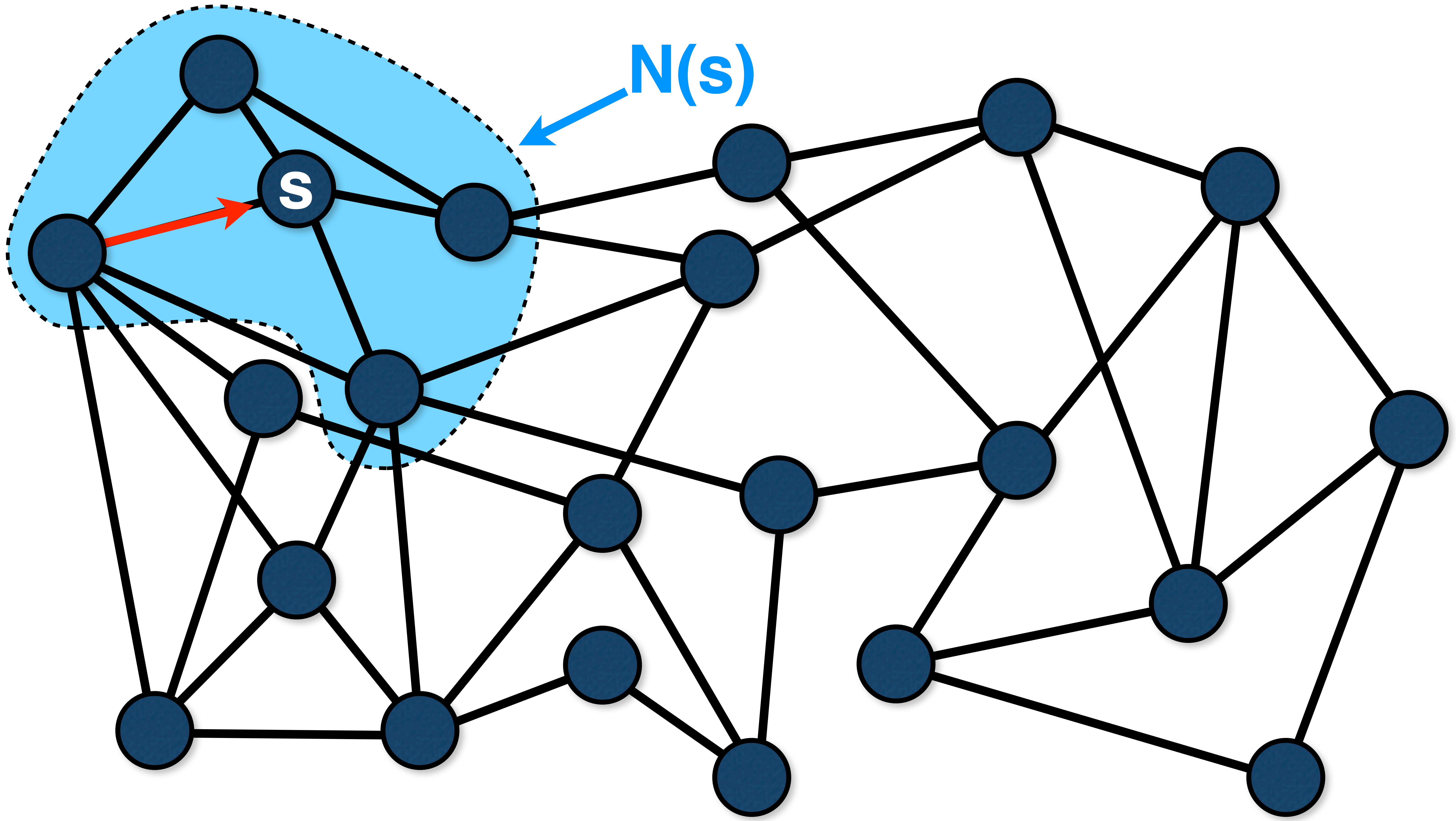
The Basic Local Search



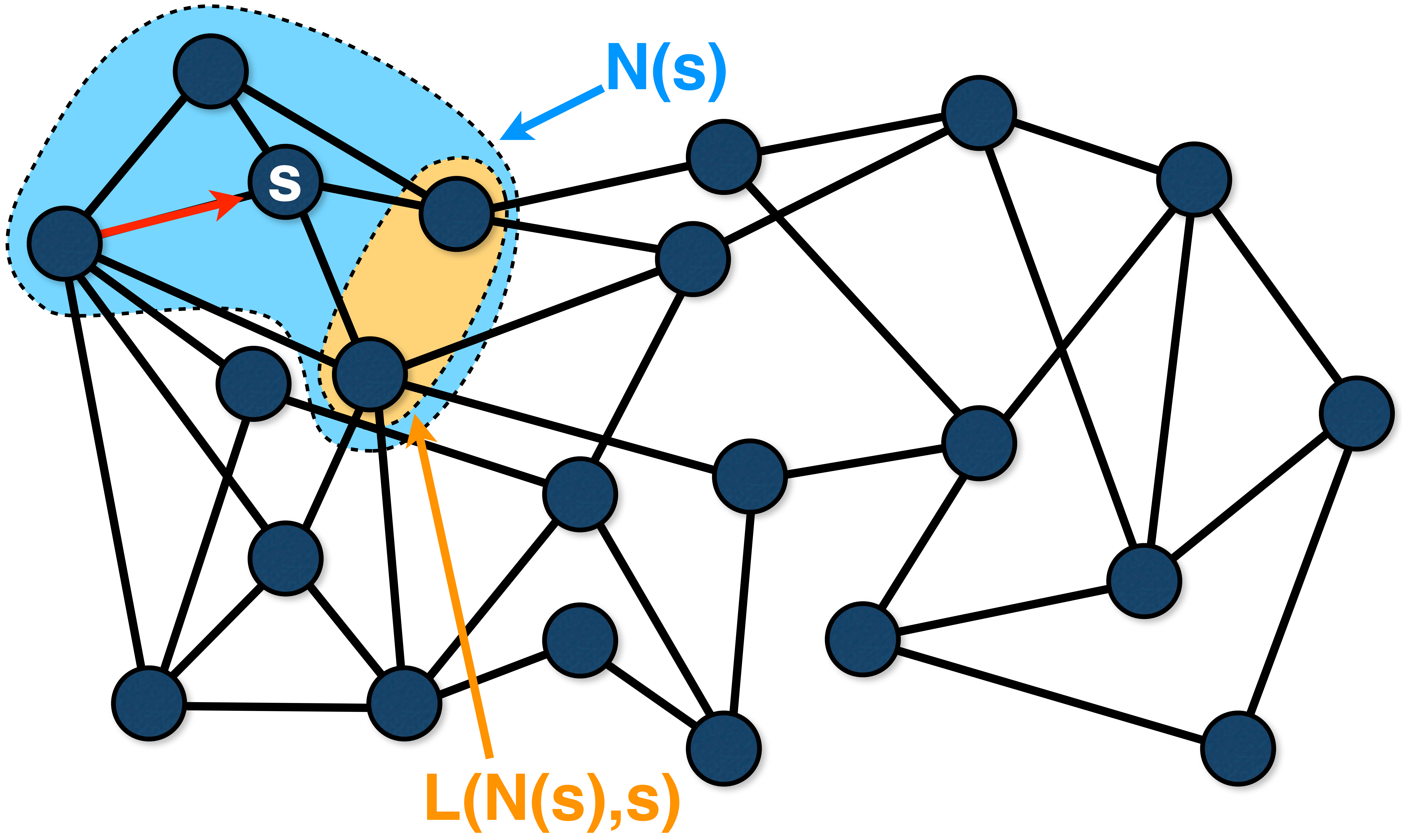
The Basic Local Search



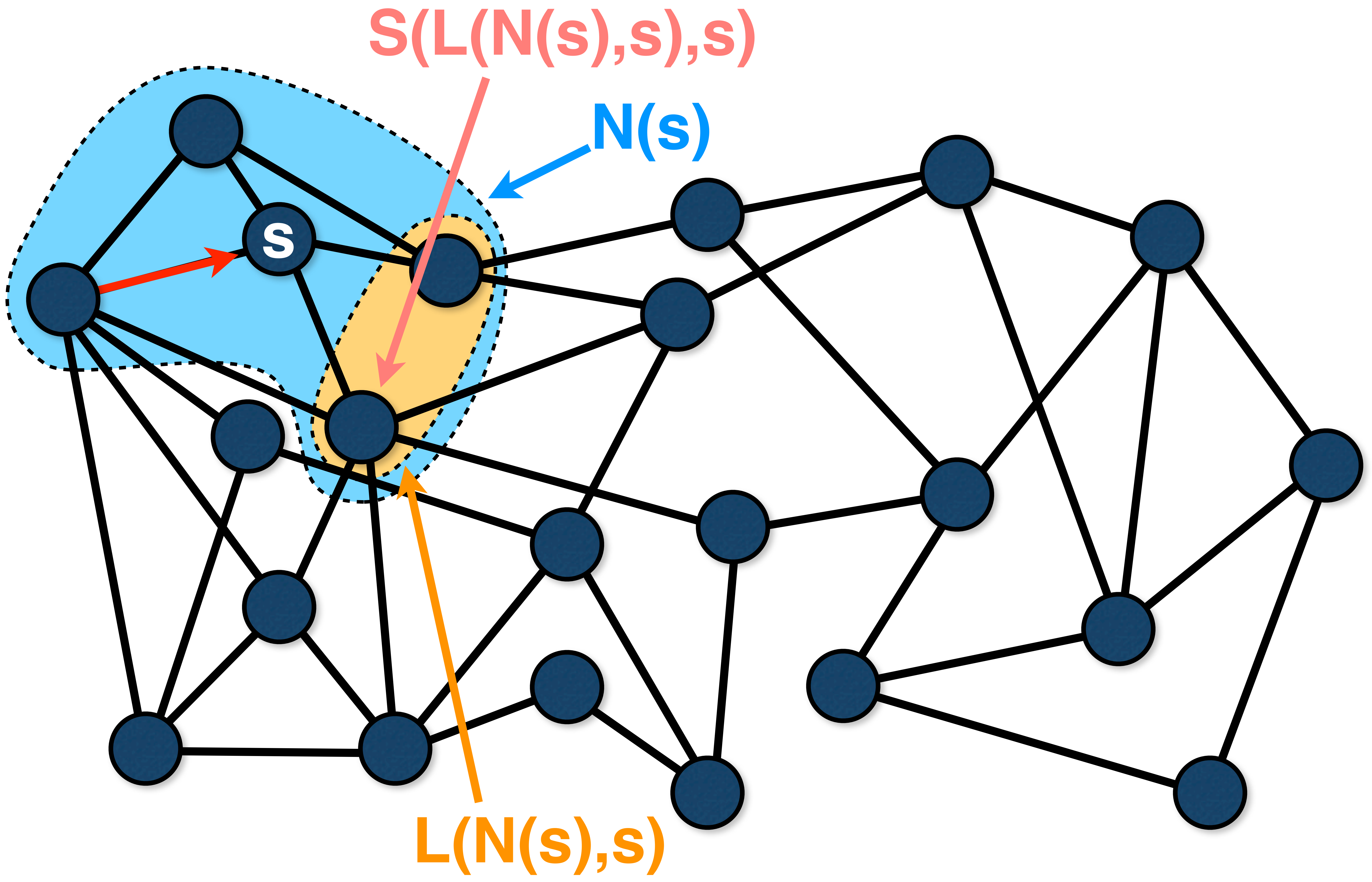
The Basic Local Search



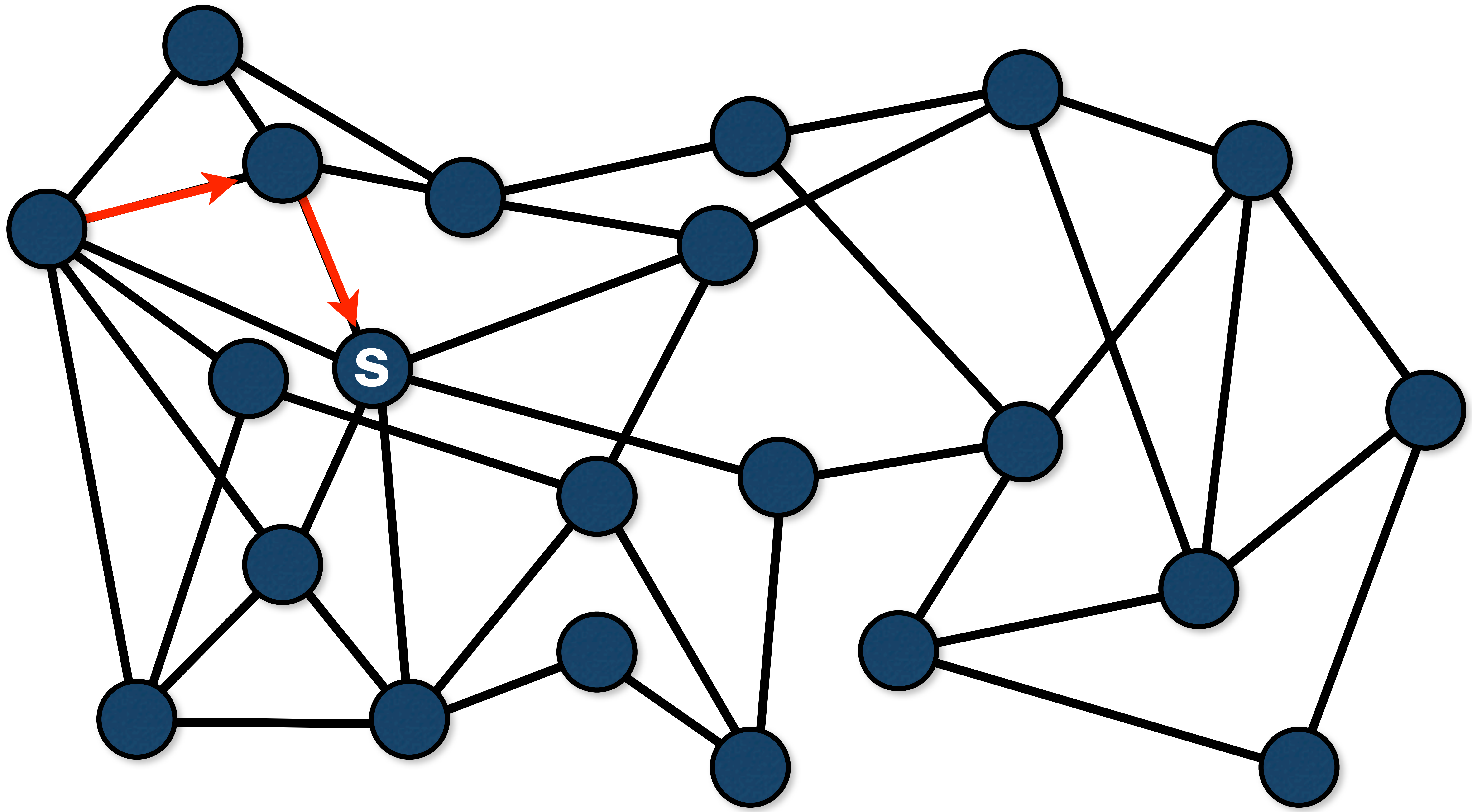
The Basic Local Search



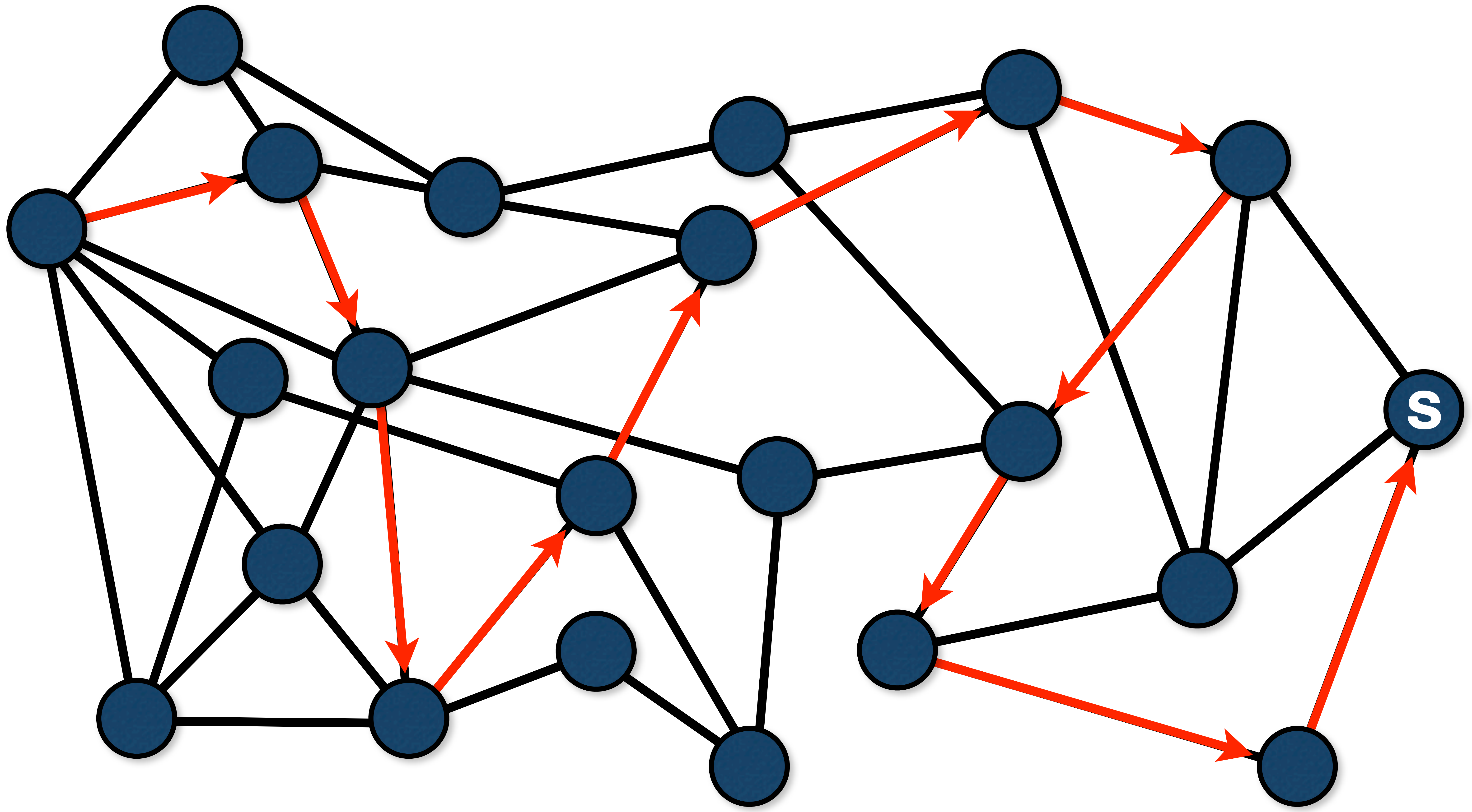
The Basic Local Search



The Basic Local Search



The Basic Local Search



A Example of Local Search

- Legal moves: local improvements
  - $L(N,s) = \{ n \text{ in } N \mid f(n) < f(s) \}$

A Example of Local Search

- Legal moves: local improvements
  - L(N,s) = { n in N I f(n) < f(s) }

- Selection function: greedy selection
  - S(L,s) = arg-min(n in L) f(n)

Heuristics and Metaheuristics

- Heuristics
  - choose the next neighbor
  - use local information:
    - the state s and its neighborhood
  - drive the search towards a local minimum

Heuristics and Metaheuristics

- Heuristics
  - choose the next neighbor
  - use local information:
    - the state s and its neighborhood
  - drive the search towards a local minimum
- Metaheuristics
  - aim at escaping local minima
  - drive the search towards a global minimum
  - typically include some memory or learning

Properties of the Neighbors

- Legal neighbors
  - Conditions on the value of the objective function

Properties of the Neighbors

- Legal neighbors
  - Conditions on the value of the objective function
- Local improvement
  - $L(N,s) = \{ n \text{ in } N \mid f(n) < f(s) \}$

Properties of the Neighbors

- Legal neighbors
  - Conditions on the value of the objective function
- Local improvement
  - $L(N,s) = \{ n \text{ in } N \mid f(n) < f(s) \}$
- No degradation
  - $L(N,s) = \{ n \text{ in } N \mid f(n) <= f(s) \}$

Properties of the Neighbors

- Legal neighbors
  - Conditions on the value of the objective function
- Local improvement
  - L(N,s) = { n in N | f(n) < f(s) }
- No degradation
  - L(N,s) = { n in N | f(n) <= f(s) }
- Potential degradation
  - L(N,s) = N

Selecting a Neighbor

- How to select the neighbor?
  - exploring the whole or part of the neighborhood

Selecting a Neighbor

- How to select the neighbor?
  - exploring the whole or part of the neighborhood
- Best neighbor
  - select “the” best neighbor in the neighborhood

Selecting a Neighbor

- How to select the neighbor?
  - exploring the whole or part of the neighborhood
- Best neighbor
  - select “the” best neighbor in the neighborhood
- First neighbor
  - select the first “legal” neighbor in the neighborhood

Selecting a Neighbor

- How to select the neighbor?
  - exploring the whole or part of the neighborhood
- Best neighbor
  - select “the” best neighbor in the neighborhood
- First neighbor
  - select the first “legal” neighbor in the neighborhood
- Multi-stage selection
  - first select one “part” of neighbor
  - second select the remaining “part” of the neighbor

Best Neighbor

Best Neighbor

- Randomization is often important in local search
  - more on this soon

Best Neighbor

- Randomization is often important in local search
  - more on this soon
- Best neighbor

```
1. **function** S-Best(N,s)
2.     $N^* := \{\ n \in N \mid f(n) = \min_{s \in N} f(s)\ \}$;
3.     **return** $n \in N^*$ with probability $1/\#N^*$;
```

Best Neighbor

- Randomization is often important in local search
  - more on this soon
- Best neighbor

```
1. function S-BEST(N,s)
2.     $N^* := \{\ n \in N \mid f(n) = \min_{s \in N} f(s)\ \}$;
3.     return $n \in N^*$ with probability $1/\#N^*$;
```

- Best improvement

```
1. function BESTIMPROVEMENT(s)
2.     return LOCALSEARCH($f$,$N$,L-IMPROVEMENT,S-BEST);
```

First Neighbor

First Neighbor

- First neighbor (in some lexicographic order)
  - avoid scanning the entire neighborhood

First Neighbor

- First neighbor (in some lexicographic order)
  - avoid scanning the entire neighborhood
- First neighbor

```
1. **function** S-FIRST(N,s)
2.     **return** $n \in N$ minimizing $lex(n)$;
```

First Neighbor

- First neighbor (in some lexicographic order)
  - avoid scanning the entire neighborhood
- First neighbor

```
1. function S-FIRST(N,s)
2.     return $n \in N$ minimizing $lex(n)$;
```

- First improvement

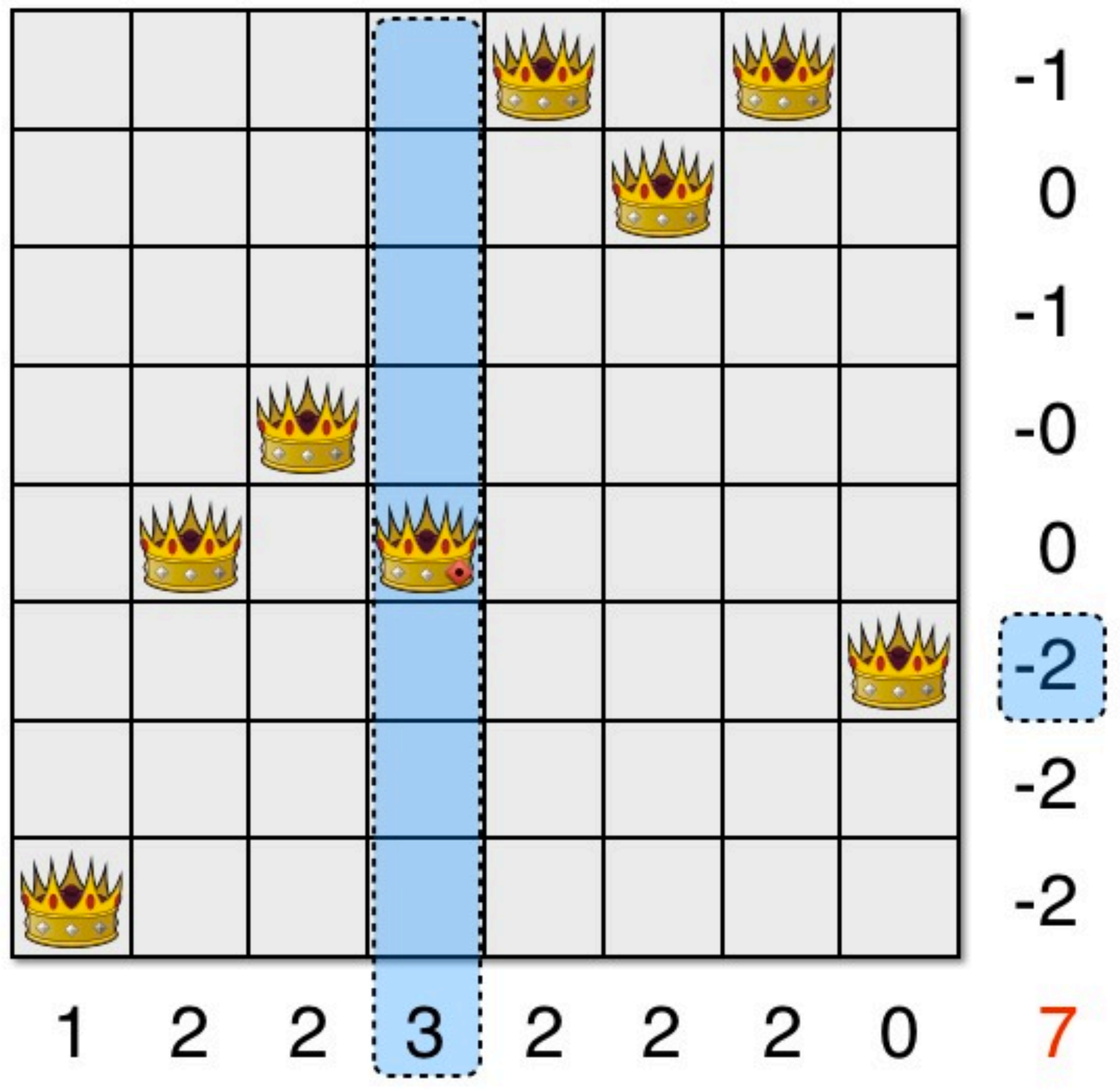
```
1. function FIRSTIMPROVEMENT(s)
2.     return LOCALSEARCH($f$,$N$,L-IMPROVEMENT,S-FIRST);
```

Multi-Stage Heuristics

- Motivation
  - avoid scanning the entire neighborhood
  - still keep a greedy flavor

Multi-Stage Heuristics

- Motivation
  - avoid scanning the entire neighborhood
  - still keep a greedy flavor



Multi-Stage Heuristics

- Max/Min-Confict
  - select the variable with the most violations
    - first stage: greedy
  - select the value with the fewest resulting violations
    - second stage: greedy

Multi-Stage Heuristics

- Max/Min-Confict
  - select the variable with the most violations
    - first stage: greedy
  - select the value with the fewest resulting violations
    - second stage: greedy
- Min-conflict heuristic
  - randomly select a variable with some violations
    - first-stage: randomized
  - select the value with the fewest resulting violations
    - second stage: greedy

Multi-Stage Heuristics

- What was the alternative?
  - N(s): { s[q ←v] | q in Queens & v in Rows }
    - s[q ←v] is the solution s where queens q is assigned to v;

Multi-Stage Heuristics

- What was the alternative?
  - N(s): { s[q ←v] I q in Queens & v in Rows }
    - s[q ←v] is the solution s where queens q is assigned to v;
- Complexity
  - quadratic: all pairs (c,r) where c is a column and r is a row
  - O(n^2) where n is the number of queens

Multi-Stage Heuristics

- What was the alternative?
  - N(s): { s[q ←v] | q in Queens & v in Rows }
    - s[q ←v] is the solution s where queens q is assigned to v;
- Complexity
  - quadratic: all pairs (c,r) where c is a column and r is a row
  - O(n^2) where n is the number of queens
- Complexity of min-conflict
  - O(n) where n is the number of queens

Multi-Stage in Car Sequencing

Slots	1	2	3	4	5	6	7	8	9	10	Demand
Class 1	🟦										1
Class 2		🟦									1
Class 3			🟦	🟦							2
Class 4					🟦	🟦					2
Class 5							🟦	🟦			2
Class 6									🟦	🟦	2

Options	1	2	3	4	5	Demand
Class 1	yes		yes	yes		1
Class 2				yes		1
Class 3		yes			yes	2
Class 4		yes		yes		2
Class 5	yes		yes			2
Class 6	yes	yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Setup	1	2	3	4	5	6	7	8	9	10	Capacity	
Option 1	🟦						✴	✴	✴	✴	1/2	3
Option 2			✴	✴	✴	✴			🟦	🟦	2/3	2
Option 3	🟦						✴	✴			1/3	2
Option 4	✴	✴			✴	✴					2/5	2
Option 5			✴	✴							1/5	3

Multi-Stage in Car Sequencing

Slots	1	2	3	4	5	6	7	8	9	10	Demand
Class 1	🟦										1
Class 2		🟦									1
Class 3			🟦	🟦							2
Class 4					🟦	🟦					2
Class 5							🟦	🟦			2
Class 6									🟦	🟦	2

Options	1	2	3	4	5	Demand
Class 1	yes		yes	yes		1
Class 2				yes		1
Class 3		yes			yes	2
Class 4		yes		yes		2
Class 5	yes		yes			2
Class 6	yes	yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Setup	1	2	3	4	5	6	7	8	9	10	Capacity	
Option 1	🟦						✖	✖	✖	✖	1/2	3
Option 2			✖	✖	✖	✖			🟦	🟦	2/3	2
Option 3	🟦						✖	✖			1/3	2
Option 4	✖	✖			✖	✖					2/5	2
Option 5			✖	✖							1/5	3

Multi-Stage Heuristics

- Quadratic neighborhood
  - consider all possible swaps
  - quadratic in the size of the assembly line

Multi-Stage Heuristics

- Quadratic neighborhood
  - consider all possible swaps
  - quadratic in the size of the assembly line
- Multi-stage neighborhood
  - select a slot s whose car induces some violations
  - swap slot s with all other slots
  - linear in the size of the assembly line

Random Walks

- Randomization
  - select a neighbor at random

Random Walks

- Randomization
  - select a neighbor at random
- Decide whether to accept it
  - random improvement
  - Metropolis algorithm

Random Walks

- Randomization
  - select a neighbor at random
- Random improvement
- Random improvement search

Random Walks

- Randomization
  - select a neighbor at random
- Random improvement

```
1. **function** S-RandomImprovement(N,s)
2.     **select** $n \in N$ with probability $1/\#N$;
3.     **if** $f(n) < f(s)$ **then**
4.         **return** $n$;
5.     **else**
6.         **return** $s$;
```

- Random improvement search

Random Walks

- Randomization
  - select a neighbor at random
- Random improvement

```
1. **function** S-RandomImprovement(N,s)
2.     **select** $n \in N$ with probability $1/\#N$;
3.     **if** $f(n) < f(s)$ **then**
4.         **return** $n$;
5.     **else**
6.         **return** $s$;
```

- Random improvement search

```
1. **function** RandomImprovement(s)
2.     **return** LocalSearch($f$,$N$,L-All,S-RandomImprovement);
```

The Traveling Tournament

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

The Traveling Tournament

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

$$d_{12} + d_{21} + d_{15} + d_{54} + d_{43} + d_{31} + d_{16} + d_{61}$$

$+ \ldots +$

$$d_{61} + d_{14} + d_{45} + d_{56} + d_{63} + d_{36} + d_{62} + d_{26}$$

The Neighborhood

- A number of moves
  - swap homes
  - swap rounds
  - swap teams
  - partial swap rounds
  - partial swap teams

Swap Teams

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	1	@3	@6	4	3	6	@4	@1	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	@2	1	5	2	@6	@3
5	@2	@3	6	4	1	@6	@4	@1	3	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	2	1	5	@2	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Swap Teams

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	2	1	5	@2	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Swap Teams

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	2	1	5	@2	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Swap Teams

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	2	1	5	@2	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

Swap Teams

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@2	4	3	@5	@4	@3	5	2	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	5	2	@1	6	@2	1	@6	@5	4
4	3	6	@1	@5	2	1	5	@2	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@5	2	@3	5	@2	3	4	1

T-R	1	2	3	4	5	6	7	8	9	10
1	6	@5	4	3	@2	@4	@3	2	5	@6
2	5	@3	6	4	1	@6	@4	@1	3	@5
3	@4	2	5	@1	6	@5	1	@6	@2	4
4	3	6	@1	@2	@5	1	2	5	@6	@3
5	@2	1	@3	@6	4	3	6	@4	@1	2
6	@1	@4	@2	5	@3	2	@5	3	4	1

Until Next Time