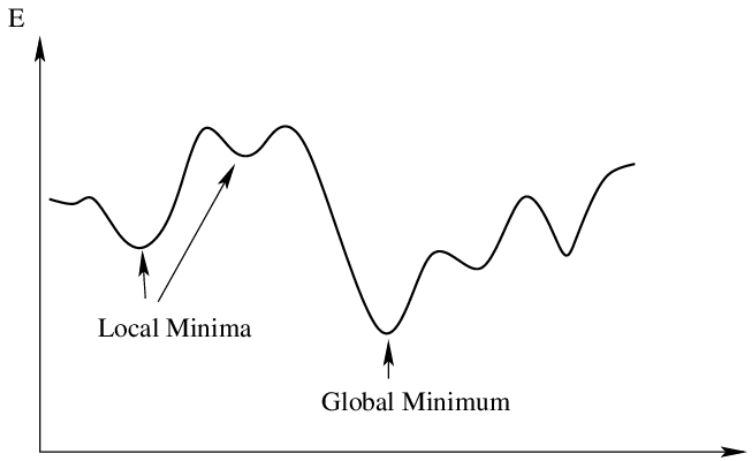
Iterated Local Search
Variable Neighborhood Search
(Vienna 2022)

Christian Blum

Artificial Intelligence Research Institute (IIIA-CSIC)



Iterated Local Search (ILS)

ILS: main info

- **In one sentence:** ILS is an algorithmic technique that iteratively employs the perturbation of the incumbent solution with a subsequent application of local search.
- **First approaches of ILS** were for the TSP [Baxter, 1981][a] and [Baum, 1986][b]
- **Formalized as a metaheuristic** by Stützle en 1998[c]

---

[a] J. Baxter (1981) Local optima avoidance in depot location. Journal of the Operational Research Society, 32, 815–819.

[b] E.B. Baum (1986) Iterated descent: A better algorithm for local search in combinatorial optimization problems. Technical report, Caltech, Pasadena, CA. manuscript.

[c] T. Stützle (1998) Local Search Algorithms for Combinatorial ProblemsAnalysis, Improvements, and New Applications. PhD thesis, Darmstadt University of Technology, Department of Computer Science.

Iterated Local Search (ILS)

The way towards ILS

ILS is the final result of different attempts to improve simple local search procedures.

A first option for improving local search

- **Iterative application of local search** applied to randomly generated initial solutions
- A sequence of independent local minima is generated in this way
- **Advantage:** theoretical guarantees can be derived
- **Disadvantage:** at some point results do hardly improve
- Does not work well especially in the context of large problem instances

Iterated Local Search (ILS)

Main feature of ILS

ILS is a metaheuristic that generates a sequence of non-independent solutions by means of the iterative application of local search.

Algorithm characteristics

- ILS is easy to understand
- Normaly easy to implement
- Basic versions often obtain excellent results

ILS — Notation

- $\mathcal{S}$: search space (set of candidate solutions)
- $s \in \mathcal{S}$: a solution
- f: objective function
- $f(s)$: objective function value of s
- s^*: local minimum
- $\mathcal{S}^*$: set of all local minima
- The used local search method defines a mapping: $\mathcal{S} \mapsto \mathcal{S}^*$

ILS: basic idea

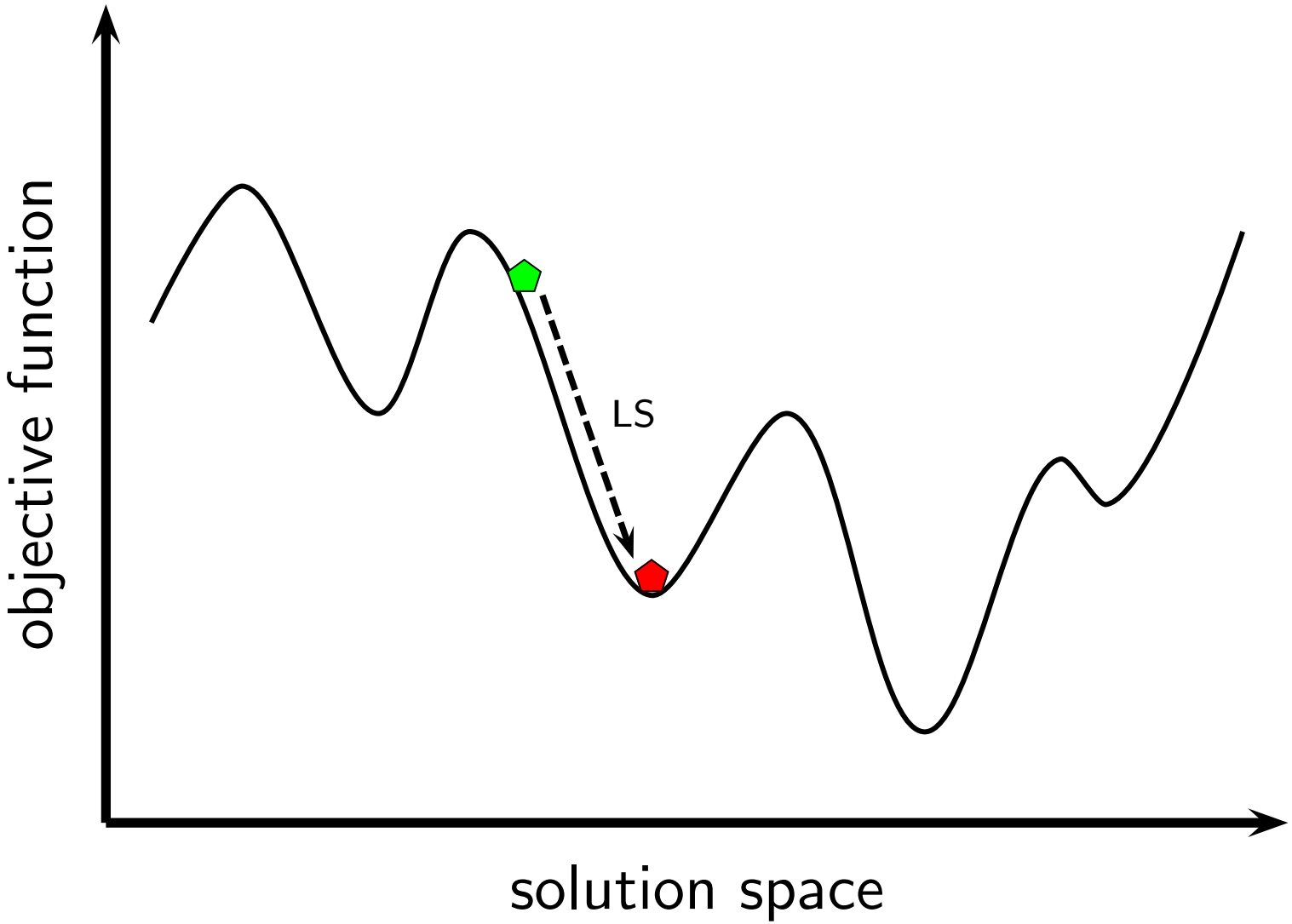
Basic idea: iterate the following instructions

- Given a solution s^*, generate a perturbation s'
- Apply local search to s', resulting in a local minimum $s^{*\prime}$
- Decide if $s^{*\prime}$ should be adopted as new incumbent solution

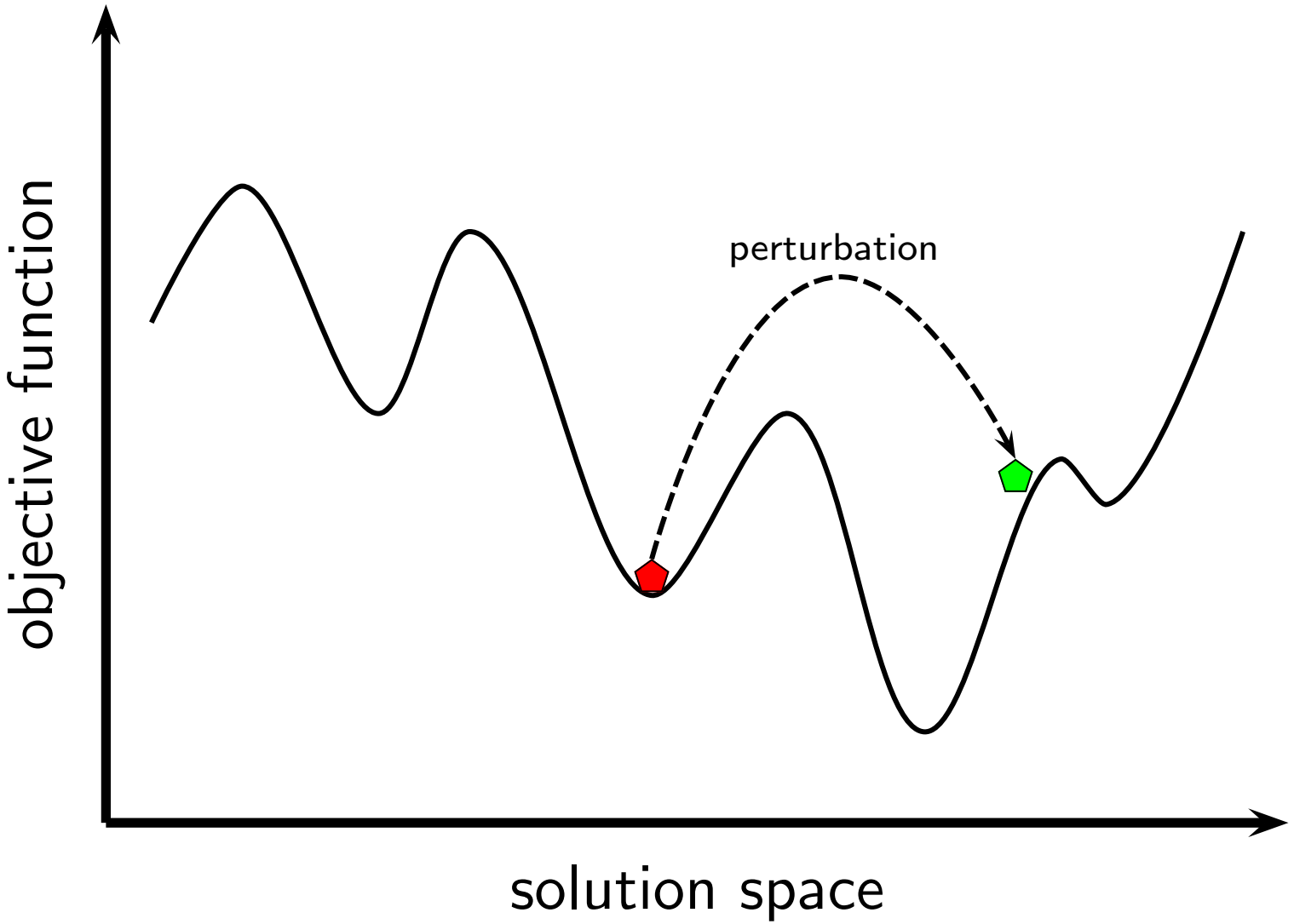
Advantage

ILS limits the search to the space $\mathcal{S}^*$ of local minima. This space is much smaller than the complete search space.

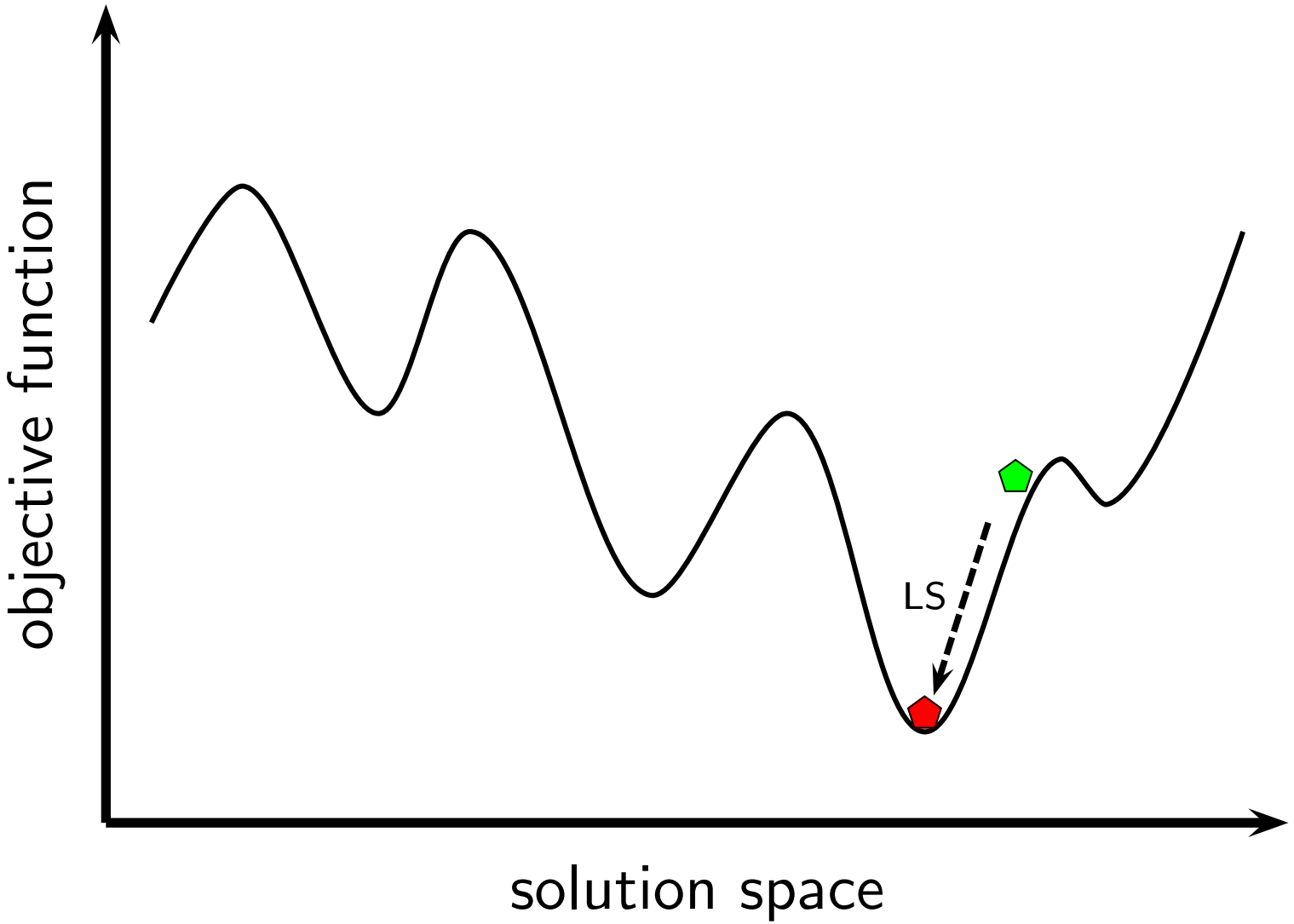
ILS: graphical illustration (1)



ILS: graphical illustration (2)



ILS: graphical illustration (3)



ILS: Pseudo code

Pseudo code

$s_0 \leftarrow$ GenerateInitialSolution()
$s^* \leftarrow$ LocalSearch(s_0)
while termination conditions not met **do**
  $s' \leftarrow$ Perturbation(s^*)
  $s^{*\prime} \leftarrow$ LocalSearch(s')
  $s^* \leftarrow$ AcceptanceCriterion(s^*,$s^{*\prime}$)
end while
output: best solution found

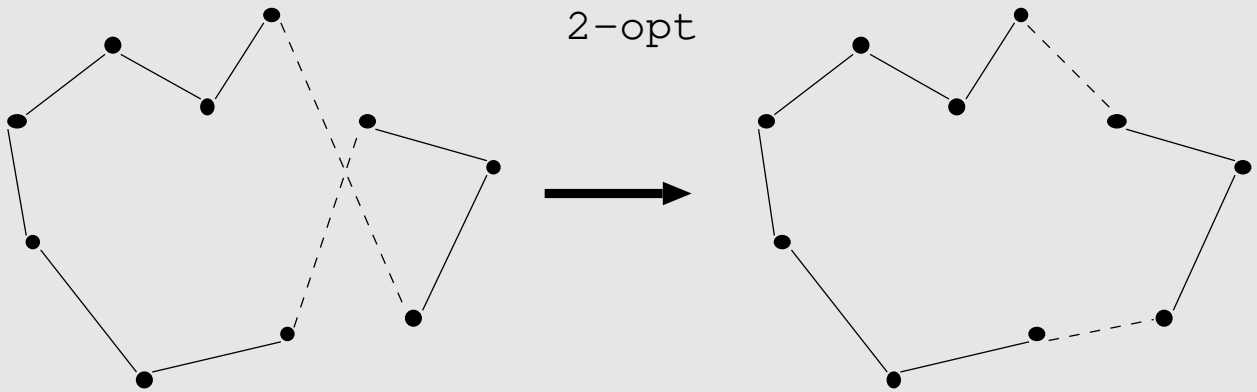
ILS: important considerations

- The performance depends on the interaction among all modules
- **A basic ILS version is easily obtained:**
  - GenerateInitialSolution: randomly or with a greedy heuristic
  - LocalSearch: choose a standard neighborhood (depending on the solution representation)
  - Perturbation: choose a neighbor from an alternative neighborhood
  - AcceptanceCriterion: only accept the new solution if it is better than the incumbent one
- Such a basic version often exhibits a very good performance
- Basic ILS versions often only require few lines of additional code

Example: TSP

Application to the TSP

- GenerateInitialSolution: *nearest neighbor* heuristic
- LocalSearch: `2-opt`, `3-opt`, `LK`(whatever is available)



- Perturbation: *double-bridge* neighborhood
- AcceptanceCriterion: accept $s^{*\prime}$ only if $f(s^{*\prime}) \leq f(s^*)$

Example: QAP

Definition: Quadratic Assignment Problem (QAP)

- **Given**: n objects and n locations
  - f_{rs}: flow from object r to object s
  - d_{ij}: distance between location i and location j
- **Goal**: find a bijection π from the n objects to the n locations that minimizes

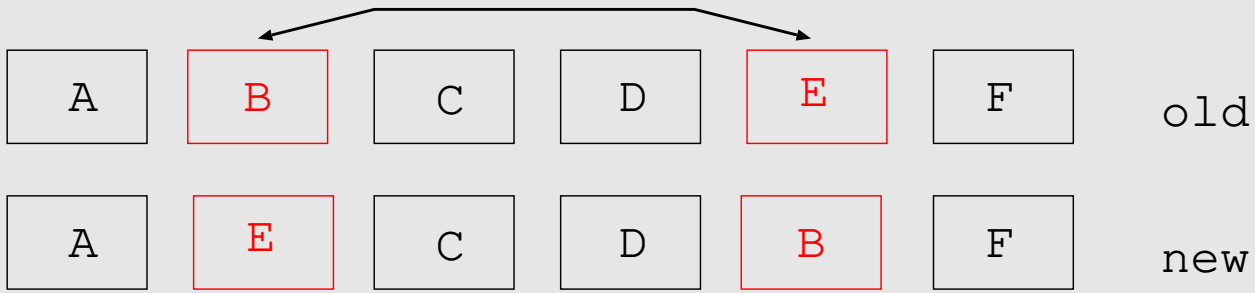
$$\min_{\pi \in \Pi(n)} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{\pi(i)\pi(j)} d_{ij}$$

  where $\pi(i)$ indicates the object at location i.
- **Info**: the QAP is among the hardest combinatorial optimization problems. It has a range of applications in industry.

Example: QAP

Application of ILS to the QAP

- GenerateInitialSolution: a random solution
- LocalSearch: local search in the 2-opt neighborhood

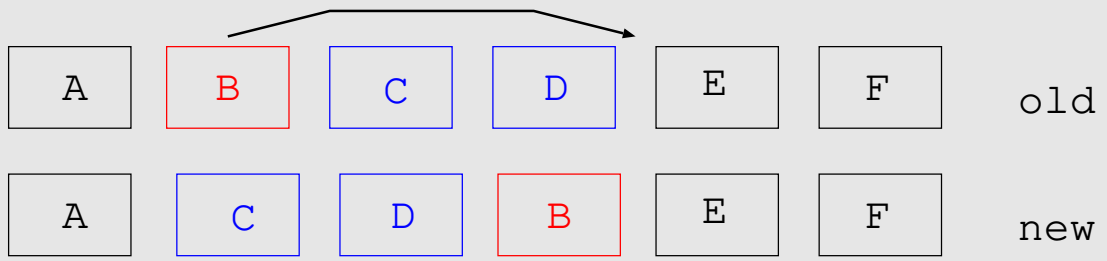


- Perturbation: choose a random solution from the k-opt neighborhood, $k > 2$
- AcceptanceCriterion: accept $s^{*\prime}$ only if $f(s^{*\prime}) \leq f(s^*)$

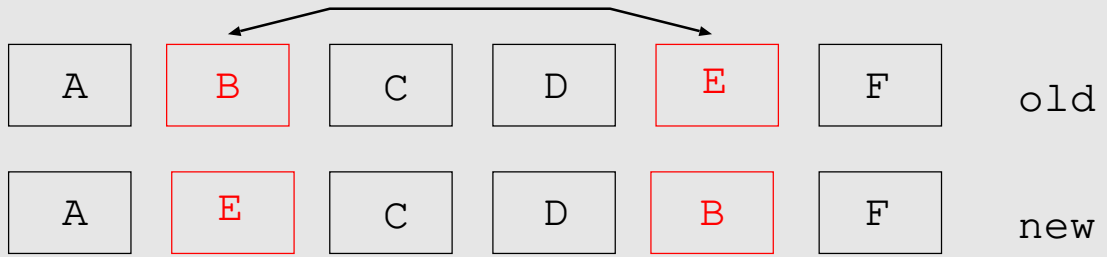
Ex.: Permutation flow shop scheduling

Application of ILS to the PFSP

- GenerateInitialSolution: heuristic by Nawatz
- LocalSearch: local search in the *insertion* neighborhood



- Perturbation: some random steps in the *interchange* neighborhood



- AcceptanceCriterion: accept $s^{*\prime}$ only if $f(s^{*\prime}) \leq f(s^*)$

ILS: algorithmic components

Optimization of individual components

- One can start simply, adding complexity step-by-step
- Normally many options are available
- **Sequential development:** optimize the individual ILS components sequentially without considering their interactions
- **Global development:** has to take into account interactions among components

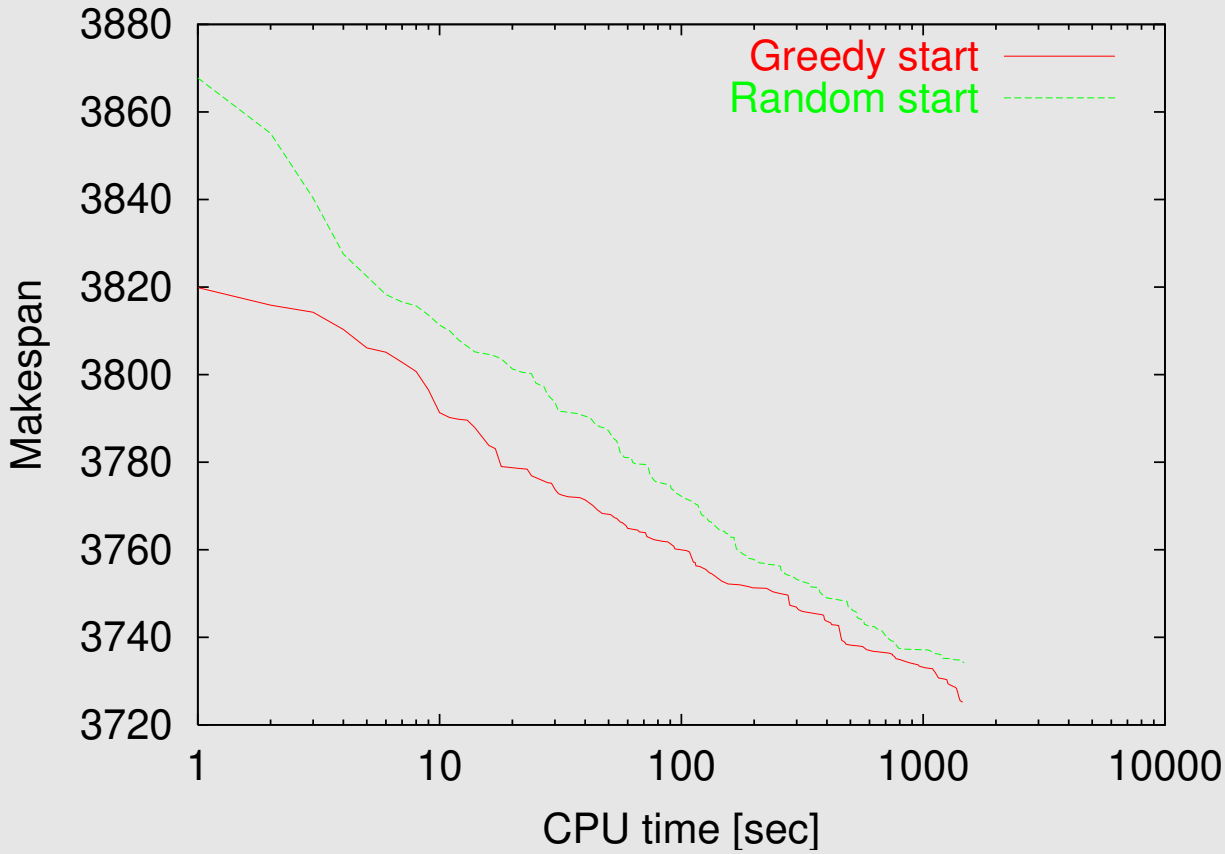
Initial solution

Considerations

- The initial solution s_0^* is the starting point of the trayectory of ILS in $\mathcal{S}^*$
- Can be generated randomly, or with a greedy heuristic
- Greedy heuristics seem to be preferable
- When a lot of computation time is available, the dependence on s_0^* is low

Influence of the initial solution

Example: PFSS problem



Perturbation

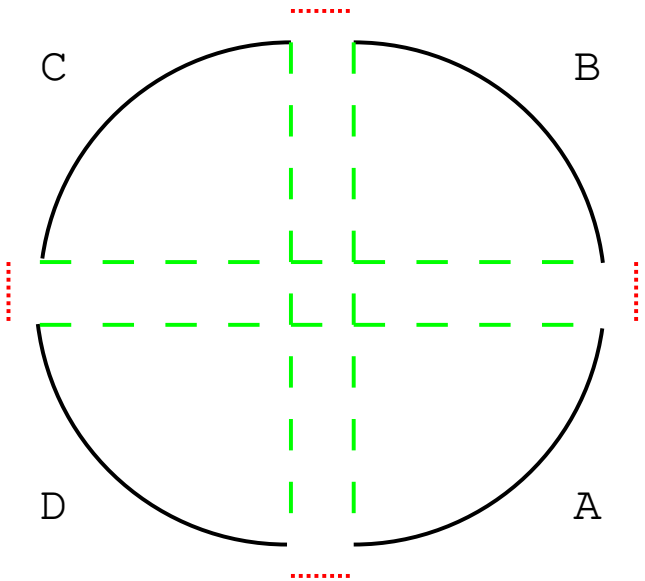
Considerations

- **Important:** the strength/degree of the perturbation
  - **Too strong**: ILS performs similar to a *multi-start* method
  - **Too weak**: ILS may stay in the same basin of attraction
- Perturbation strength may vary over the run-time (depending on the search history)
- Perturbation should be complementary to LocalSearch

Ex.: *double-bridge* perturbation (TSP)

Characteristics

- Perturbation strength is rather low
- It is complementary to *Lin-Kernighan* local search
- Does not increase the objective function value too much



Perturbation strength

Exception

- ILS for the QAP requires a **strong perturbation**
- The table shows the average deviation from best-known solutions for different sizes of the perturbation (from 3 to n); averages over 10 trials.

instance	3	$n/12$	$n/6$	$n/4$	$n/3$	$n/2$	$3n/4$	n
kra30a	2.51	2.51	2.04	1.06	0.83	0.42	0.0	0.77
sko64	0.65	1.04	0.50	0.37	0.29	0.29	0.82	0.93
tai60a	2.31	2.24	1.91	1.71	1.86	2.94	3.13	3.18
tai60b	2.44	0.97	0.67	0.96	0.82	0.50	0.14	0.43

Option: using an adaptive perturbation strength

Perturbation strength can be varied during the execution of the algorithm depending on the search process.

Perturbation vs. number of iterations

Note

The lower the perturbation strength, the more iterations can be performed by the algorithm as local search normally finishes earlier.

Example: TSP

instance	$\#_{LS_{RR}}$	$\#_{LS_{1\text{-}DB}}$	$\#_{LS_{5\text{-}DB}}$
kroA100	17507	56186	34451
d198	7715	36849	16454
lin318	4271	25540	9430
pcb442	4394	40509	12880
rat783	1340	21937	4631
pr1002	910	17894	3345
d1291	835	23842	4312
fl1577	742	22438	3915
pr2392	216	15324	1777
pcb3038	121	13323	1232
fl3795	134	14478	1773
rl5915	34	8820	556

Acceptance criterion

- The utilized AcceptanceCriterion has a strong influence on the trayectory of ILS in $\mathcal{S}^*$
- It controls the balance between intensification and diversification
- **Extreme intensification:**
  $\texttt{Better}(s^*, s^{*\prime})$: accept $s^{*\prime}$ only if $f(s^{*\prime}) < f(s^*)$
- **Extreme diversification:**
  $\texttt{Always}(s^*, s^{*\prime})$: accept $s^{*\prime}$ always
- **However:** there are many intermediate options

Acceptance criterion: TSP example

Characteristics

- It has shown that weak perturbations are sufficient
- High-quality solutions are found in clusters in the search space
  $\rightsquigarrow$ **strong intensification** is required

Observations

- **With little computation time available:** `Better` is to be preferred
- **With lots of computation time available:** more diversification is better

Local search

Options

- **Basic case:** standard local search (*best- or first-improvement*)
- **More sophisticated options:** other metaheuristics
  - Tabu search (TS)
  - Simulated annealing (SA)
  - Guided local search (GLS)

Quality of the local search

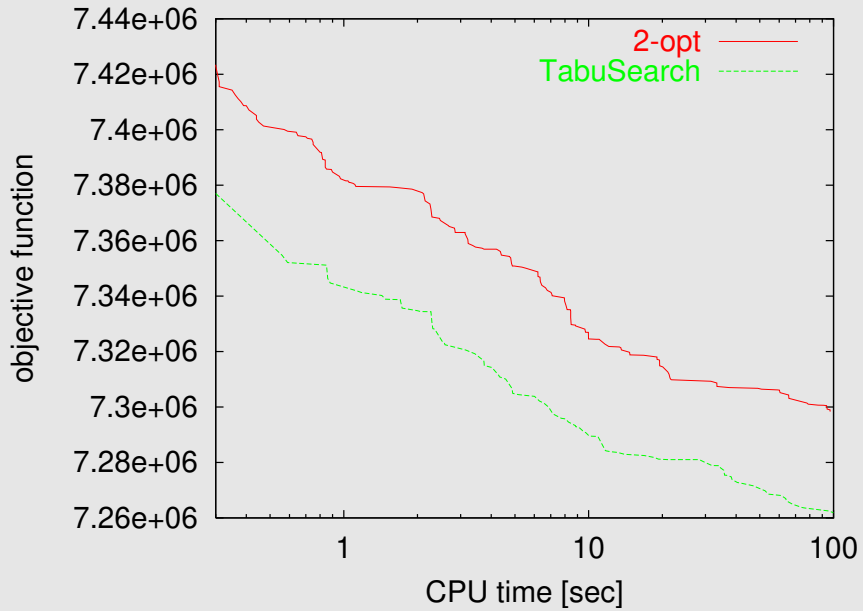
- **Generally**: the better local search, the better ILS
- The balance between speed and quality in the utilized local search method is an important aspect for the development of ILS

Local search: QAP example (1)

Experimental setup

One algorithm version uses tabu search (for $6n$ steps at each iteration). The other algorithm uses 2-opt local search. Both use the same computation time limit.

Performance for instance `tai60a (random, unstructured instance)`

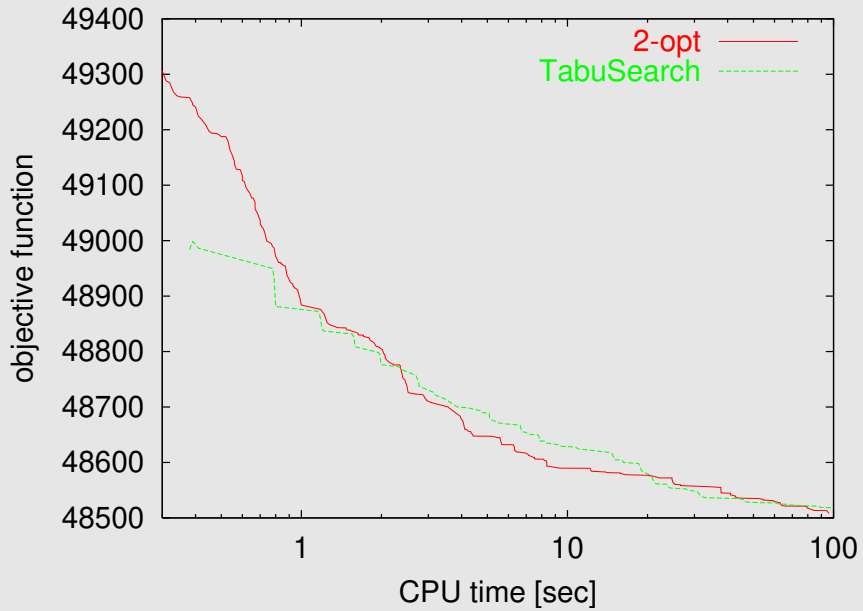


Local search: QAP example (2)

Experimental setup

One algorithm version uses tabu search (for $6n$ steps at each iteration). The other algorithm uses 2-opt local search. Both use the same computation time limit.

Performance for instance `sko64 (grid distances, structured flows)`

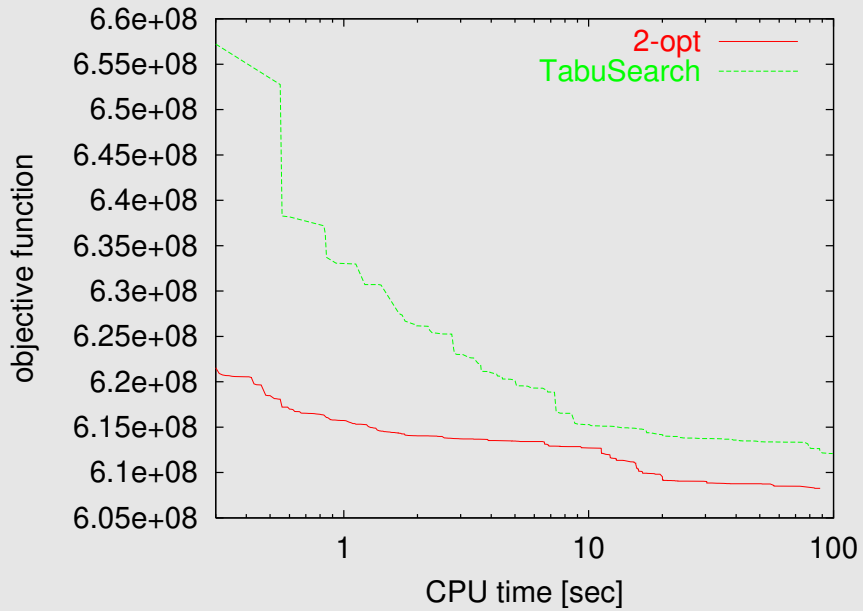


Local search: QAP example (3)

Experimental setup

One algorithm version uses tabu search (for $6n$ steps at each iteration). The other algorithm uses 2-opt local search. Both use the same computation time limit.

Performance for instance `tai60b (random, structured instance)`



Development of ILS: guidelines

Development guidelines

- The generation of the initial solution is, to a large extent, irrelevant for longer algorithm runs
- Local search should be of high quality, but the computation time should not be too high
- The choice of the perturbation method and strength is highly dependent on the chosen local search method
- The choice of the acceptance criterion depends strongly on the chosen perturbation and local search.
- Of special importance can be the interactions between the perturbation strength and the acceptance criterion.

Development of an ILS

Recommendations

- The perturbation mechanism should move the search process to other basins of attraction
- The combination between the perturbation and the acceptance criterion determines the relative balance between intensification and diversification; large perturbations are only useful if they can be accepted.

In general

A good balance between intensification and diversification is important and hard to achieve

Questions?

Variable Neighborhood Search (VNS)

Essential information

- **In one sentence:** VNS is a metaheuristic that is based on the systematic change of the neighborhood during the search.
- **One of the first applications** was the one to the *p-median* problem [Hansen and Mladenović, 1997][a]
- **Formalized as a metaheuristics by** Mladenović and Hansen en 1997[b]

---

[a] P. Hansen, N. Mladenović, **Variable neighborhood search for the *p*-median problem**, *Location Science*, 5(4), pages 207–226, 1997

[b] N. Mladenović and P. Hansen, **Variable neighborhood search**, *Computers & Operations Research*, 24, pages 1097–1100, 1997

VNS: main idea (1)

Observe

VNS and ILS are very much related.

What has led to the development of VNS?

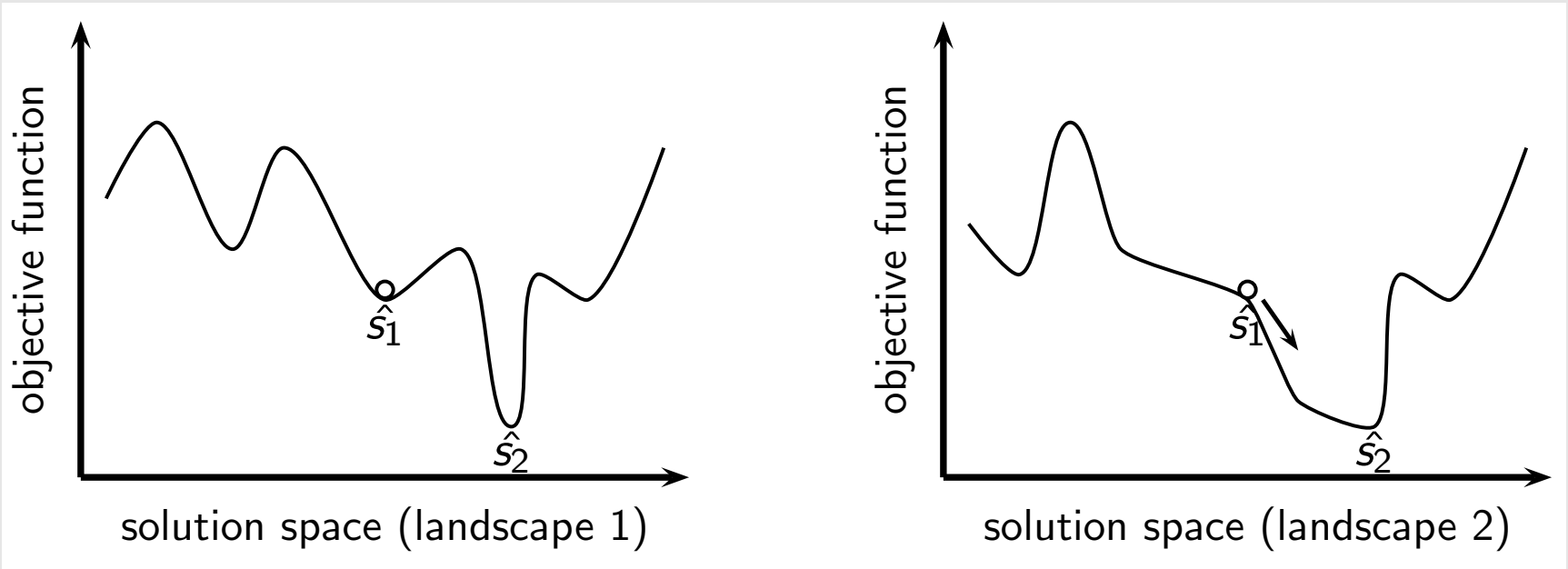
- **Observation 1:** A local minimum w.r.t. a neighborhood structure N_1 is not necessarily locally minimal w.r.t. another neighborhood structure N_2.
- **Observation 2:** A global optimum is locally optimal w.r.t. all neighborhood structures.

Main idea

Switch between different neighborhoods during the search.

VNS: main idea (2)

Main idea: graphical illustration



VNS: variants

Some well-known VNS variants

- Variable neighborhood descent (VND)
- Basic variable neighborhood search (VNS)
- Reduced variable neighborhood search (RVNS)
- Variable neighborhood decomposition search (VNDS)

Typical notations

- $\mathcal{N}_k, k = 1, \ldots, k_{max}$ is a set of k neighborhoods
- $\mathcal{N}_k(s)$ refers to the set of neighbors of s in the k-th neighborhood $\mathcal{N}_k$.

Sets of neighborhoods: examples

k-exchange neighborhoods

- **In subset selection problems:** remove k elements from a solution. Subsequently add k different elements to the resulting partial solution.
- **In permutation problems:** k-opt neighborhoods

Neighborhoods based on distance measures

- **Bit strings:** the neighbors of a solution are all those strings with a *Hamming* distance of k to the solution
- **...**

Variable neighborhood descent (VND)

Note

VND is a simple (but clever) extension of local search.

Pseudo code

Choose a set $\mathcal{N}_k$, $k = 1, \ldots, k_{\max}$, of neighborhoods
$s \leftarrow$ GenerateInitialSolution()
$k \leftarrow 1$
while $k \leq k_{\max}$ **do**
  $s' \leftarrow$ ChooseBestNeighbor($\mathcal{N}_k(s)$)
  if $f(s') < f(s)$ **then**
    $s \leftarrow s'$, $k \leftarrow 1$
  else
    $k \leftarrow k + 1$
  end if
end while

VND: additional information

- The final solution is a local minimum with respect to all neighborhoods N_k, $k = 1, \ldots, k_{\max}$
- Typically, neighborhoods are ordered from smallest to largest
- In case $\mathcal{N}_k, k = 1, \ldots, k_{\max}$ are black-box neighborhood functions:
  - Order the neighborhood function in some way
  - Apply them in the determined order
- **Advantage of VND:** able to provide very good solutions in little computational time

Pseudo code

Choose a set $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, of neighborhoods
$s \leftarrow$ GenerateInitialSolution()
while termination conditions not met **do**
  $k \leftarrow 1$
  while $k \leq k_{max}$ **do**
    $s' \leftarrow$ ChooseRandomNeighbor($\mathcal{N}_k(s)$) {*Shaking* phase}
    $s'' \leftarrow$ LocalSearch(s')
    if $f(s'') < f(s)$ **then** $s \leftarrow s''$, $k \leftarrow 1$
    else $k \leftarrow k + 1$ **end if**
  end while
end while

Characteristics of basic VNS

- The local search procedure LocalSearch() makes use of a standard neighborhood
- The other neighborhoods are explored randomly by function ChooseRandomNeighbor($\mathcal{N}_k(s)$)
- **Note:** The *shaking* phase corresponds to the *perturbation* in ILS
- The degree of perturbation (as performed by shaking) is systematically varied during the search.
- Acceptance criterion: a new solution s'' is only accepted if it is better than the incumbent solution s

- **Order in which neighborhoods are explored:**
  - **forward VNS:** starts with $k = 1$ and increases k by one if no better solution is found; otherwise $k \leftarrow 1$
  - **backward VNS:** starts with $k \leftarrow k_{max}$ and decreases k by one if no better solution is found
- **Acceptance of solutions worse than the incumbent:**
  - Accept the new solution s'' with some probability if it is worse than s
  - **Skewed VNS:** accept s'' if

$$f(s'') - \alpha d(s, s'') < f(s)$$

where $d(s, s'')$ measures the distance between solutions

Reduced VNS

Reduced VNS: features

- The same as basic VNS except that no LocalSearch() procedure is applied
- Limited to exploring randomly the different neighborhoods
- Especially useful for large-scale problem instances for which standard local search takes a lot of computation time

Recent article

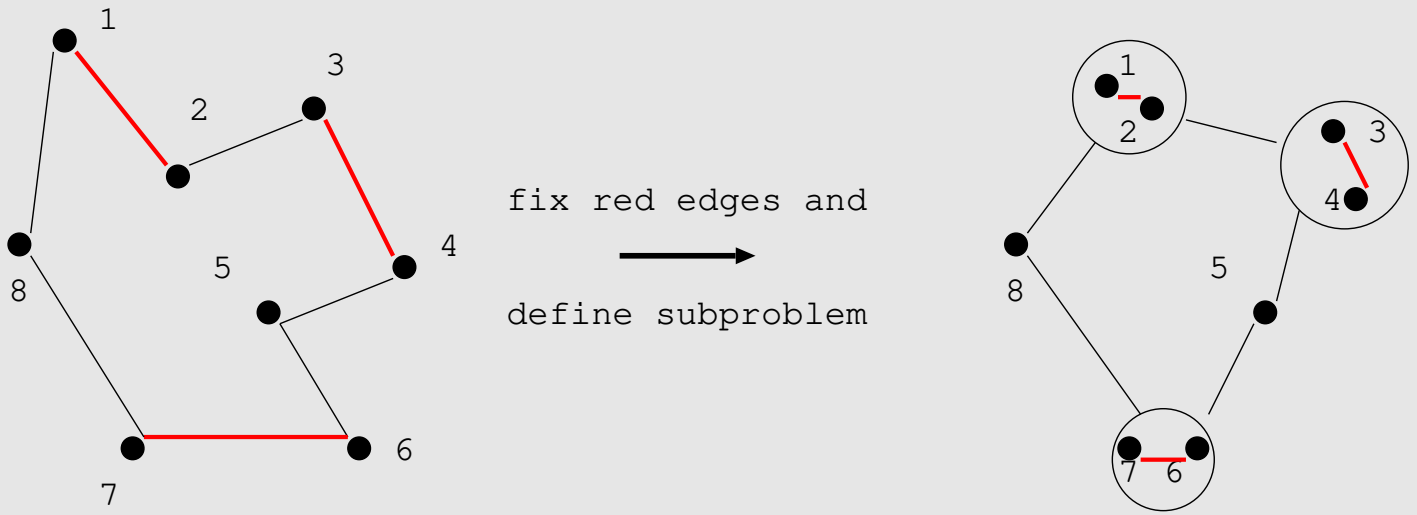
Cheimanoff, N., Fontane, F., Kitri, M. N., & Tchernev, N. (2021). *A reduced VNS based approach for the dynamic continuous berth allocation problem in bulk terminals with tidal constraints*. **Expert Systems with Applications**, 168, 114215.

Variable neighb. decomposition search

Central idea

- Generates subproblems by keeping all but k solution components fixed
- Applies local search only to the k free components

Graphical illustration



Pseudo code

Select a set of neighborhood structures: $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$
$s \leftarrow$ GenerateInitialSolution()
while termination conditions not met **do**
  $k \leftarrow 1$
  while $k \leq k_{max}$ **do**
    $s' \leftarrow$ PickAtRandom($\mathcal{N}_k(s)$) {s and s' differ in k attributes}
    $s'' \leftarrow$ LocalSearch(s') {only moves involving the k free attributes}
    if $f(s'') < f(s)$ **then** $s \leftarrow s''$, $k \leftarrow 1$
    else $k \leftarrow k + 1$ **end if**
  end while
end while

Relationship between ILS and VNS

- The two metaheuristic methods are based on different underlying philosophies
- They are similar in many aspects
- ILS appears to be more flexible with respect to the optimization of the interaction of algorithmic components
- Generally they are both robust metaheuristics
- In general both are highly efficient

Questions?