Section 2

Neighborhood-Based Approaches

Section 2.1: Local Search

Neighborhoods

- Consider an instance I of an optimization problem. Let F_I be the set of feasible solutions and $C_I : F_I \mapsto \mathbb{R}$ be the objective function.
- Given a feasible point $f \in F_I$, it is useful to define a set $N_I(f)$ of points that are "close" in some sense to the point f.
- A *neighborhood* is a mapping $N_I : F_I \mapsto 2^{F_I}$ from the set of feasible points to its power set.
- In principle, the mapping can be chosen arbitrarily, but of course depending of the problem only certain neighborhoods will be useful.
- Later on, we will discuss some general rules of thumb how to construct neighborhoods.

Neighborhood Graphs

- A given neighborhood $N_I(\cdot)$ for an instance I of an optimization problem induces a *neighborhood graph* $G_I = (V_I, A_I)$.
- The vertex set V_I corresponds to the set of feasible points F_I.
- There is an arc $(x, y) \in A_I$ if and only if $y \in N_I(x)$.
- *Note*: In case of problems with potentially infinite solution spaces, these graphs may be of infinite size, and the number of arcs leaving/entering a node may also be infinite.
- The neighborhood graph can be considered as an undirected graph if the neighborhood is *symmetric*, i.e. if $y \in N_I(x) \Leftrightarrow x \in N_I(y)$ for all $x, y \in V_I$.

Global vs. Local Optima

- Let $I \in \mathcal{I}$ be an instance:
  - A *global minimum* for I is a feasible solution $s^* \in F_I$ such that $C_I(s) \geq C_I(s^*)$ for all $s \in F_I$.
  - A *local minimum* for I is a feasible solution $s^* \in F_I$ such that $C_I(s) \geq C_I(s^*)$ for all $s \in F_I$ with $(s^*, s) \in A_I$.
- A neighborhood relation is called *exact* (with respect to a minimization problem) if every local minimum is also a global minimum.
- Local/global *maxima* and exactness with respect to a *maximization* problem are analogously defined.

General Local–Search Scheme

If terminating, the following general algorithmic scheme obviously delivers a local minimum:

- **Step 1:** Start with an arbitrary feasible solution $s^* \in F_I$.
- **Step 2:** While there is some $s \in F_I$ with $(s^*, s) \in A_I$ and $c(s) < c(s^*)$:
  a) Select one such s.
  b) Let s be the new s^*.
- **Step 3:** Deliver s^* as the final solution.

Comments:

- In case of an exact neighborhood, this clearly means that this algorithmic scheme solves the minimization problem optimally (if terminating!).
- In case F_I is finite, the algorithm certainly terminates after a finite number of iterations.

Examples of Neighborhood Relations

Before we start with examples:

- In the general definition from the previous slides, a neighborhood relation is an arbitrary graph on the solution space of an instance.
- In practice, the number of arcs leaving or entering a given solution is typically tiny compared to the size of the solution space.
- Since the solution space is usually huge, this does not mean that the number of arcs leaving/entering a node is small in absolute terms.
- Fortunately, we usually do not need to construct (or even store) the whole neighborhood graph explicitely.
- Typically, the existence of an arc (s_1, s_2) in A_I is constituted by minor modifications, which transform s_1 into s_2.

  $\longrightarrow$ In the following examples, we only formulate these minor modifications to specify A_I.

Example I: Permutations

- The feasible outputs of various problems may be interpreted as permutations of a common ground set $\{1, \ldots, n\}$.

- *Examples*:
  - sorting,
  - TSP.

Natural options for neighborhood relations (examples only!):

Two permutations $\sigma_1 \neq \sigma_2$ of $\{1, \ldots, n\}$ are neighbored if they are identical except for:

- a single swap: (one pair of elements is exchanged):

$$\exists\, i, j \in \{1, \ldots, n\},\, i \neq j,\, \forall\, k \in \{1, \ldots, n\} \setminus \{i, j\} : \sigma_1[k] = \sigma_2[k]\,.$$

  $\longrightarrow$ Symmetric neighborhood relation.

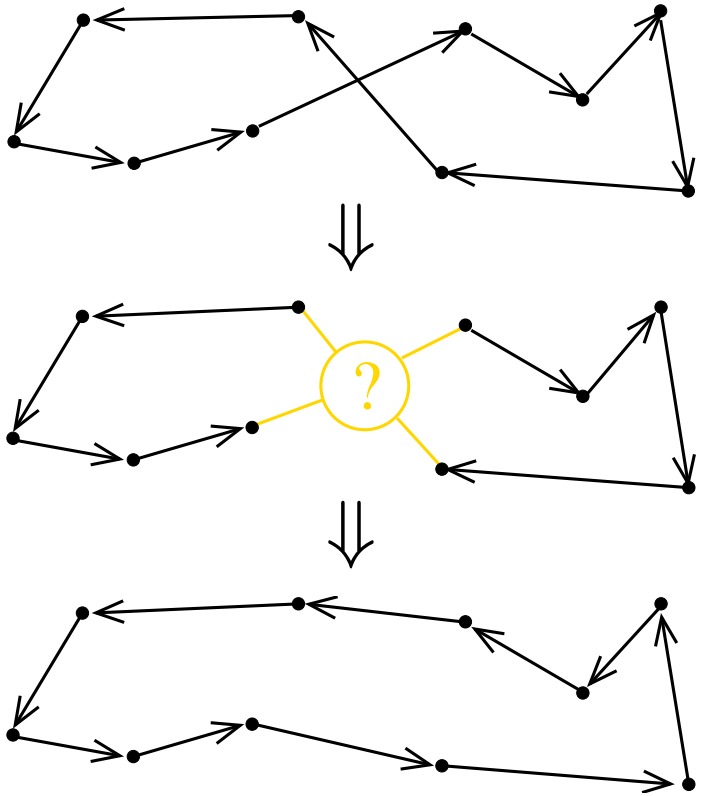
- The permutations σ_1 and σ_2 differ by a rotational shift:

$$\begin{aligned}
&\exists\, i, j \in \{1, \ldots, n\},\, i < j,\\
&\qquad (\forall\, k \in \{1, \ldots, i-1,\, j+1, \ldots, n\} : \sigma_1[k] = \sigma_2[k]) \;\wedge\\
&\qquad (\forall\, k \in \{i, \ldots, j-1\} : \sigma_1[k+1] = \sigma_2[k]) \;\wedge\\
&\qquad (\sigma_1[i] = \sigma_2[j])\,.
\end{aligned}$$

  $\longrightarrow$ Asymmetric neighborhood relation.

Example II: (Euclidean) TSP

Remove two arcs, re–connect the tour by inserting two new arcs, and turn some of the arcs to make the tour properly oriented again.

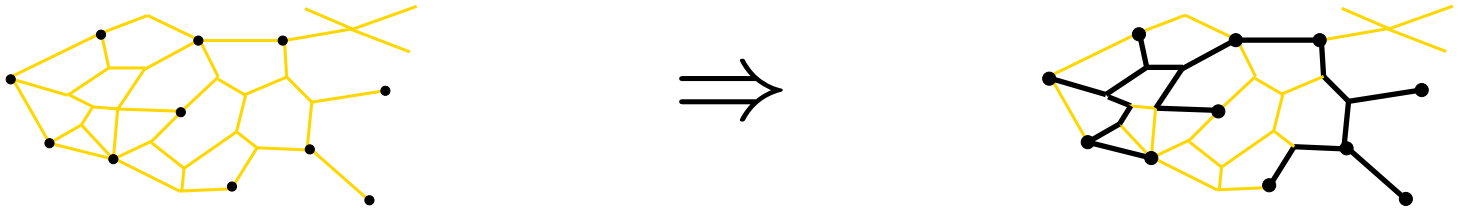


k-opt-Neighborhood for the TSP

- In the simple neighborhood definition on the last slide, *two* arcs were exchanged.
- *Obvious generalization*:
  - a fixed number $k \geq 2$ of arcs is removed;
  - k appropriate arcs are introduced to re–connect the partial tours;
  - an appropriate selection of these partial tours is turned in order to make the re–connected subgraph an oriented tour.
- This kind of neighborhood is called k–*OPT* in the literature.
- Consequently, the simple neighborhood from the last slide is "2–OPT".
- The size of the k-opt neighborhood is $\Omega(n^k)$ for the TSP on n points.
- Therefore, in practice, only 2–OPT (or 3–OPT) are usually applied.

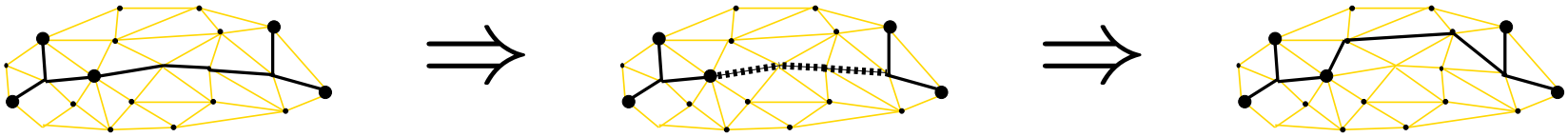
Example III: Steiner Trees in Networks

Recall the Steiner tree problem:



Example of a neighborhood structure for Steiner trees:

- Remove a complete path between two nodes of interest (that is, branchings of the tree or terminals),
- then re–connect the two resulting connected components by some new path.



Example IV: Bipartion

- *Problem definition*:
  - *Input*: positive real numbers $a[1], \ldots, a[n]$, b.

    $\longrightarrow$ Items $1, \ldots, n$ with sizes $a[1, \ldots, n]$; capacity b.

  - *Feasible outputs*: selections

    $$\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$$

    of arbitrary sizes k such that $a[i_1] + \cdots + a[i_k] \leq b$.

  - *Objective*: maximizing $a[i_1] + \cdots + a[i_k]$.

- *Straightforward idea* for a neighborhood structure:

  Two feasible solutions are neighbored.

  $\Longleftrightarrow$

  One can be constructed from the other one by exchanging a single item.

Bipartition (cont'd):

- *Problem*: The number of items in the selection cannot change by stepping from one feasible solution to a neighbored one.

  $\longrightarrow$ G_I is highly disconnected.

  $\longrightarrow$ If the search happens to start in the "wrong" connected component, it has no chance to reach the good solutions.

- *Probably a better approach*:

  Two feasible solutions are neighbored.
  $\Longleftrightarrow$
  One can be constructed from the other one by inserting, removing, or exchanging one item.

- In general:
  It is desirable that the neighborhood graph is *strongly connected* (i.e. there is a directed path between any two vertices).

Example V: Disjoint Paths

- *Input*:
  - a directed or undirected graph,
  - a set of k pairs of nodes,
  - natural numbers $\ell_1, \ldots, \ell_k$.
- Pairs of nodes:
  - ordered pairs $(s_1, t_1), \ldots, (s_k, t_k)$ in case of directed graphs,
  - unordered pairs $\{s_1, t_1\}, \ldots, \{s_k, t_k\}$ in case of undirected graphs.
- *Feasible output*: a set of paths such that
  - every path runs from some s_i to its mate t_i,
  - at least ℓ_i paths run from s_i to t_i, and
  - all paths altogether fulfill some *disjointness condition*.

Disjointness conditions:

- *Edge/arc–disjointness*: No two paths share an edge/arc.
- *Node–disjointness*: No two paths share a node except for, possibly, common endnodes (*internally* node–disjoint).
- *Capacities*:
  - *Additional input*: Each edge/arc is assigned a nonnegative integral *capacity value*.
  - Additional side constraint for each edge/arc:

    The number of paths using this edge/arc must not exceed this edge's/arc's capacity value.

Neighborhood for Disjoint Paths?

Discussion:

- This is not a good example of neighborhood structures. In fact, it was inserted into this list of examples to serve as a *counter*–example.
- *What's wrong*:
  Seems that there is no appropriate neighborhood structure.
- *Straightforward ideas* for neighborhood structures: two feasible solution are neighbored $\Longleftrightarrow$
  - a subpath of one path is changed, or
  - subpaths of a few, mutually involved paths are exchanged, or
  - some edges change the path to which they belong.
- *Why not too promising*:
  G_I is usually "torn" into very small connected components.

  $\longrightarrow$ Chances are high that the local search is over after very few (maybe zero) steps.

  $\longrightarrow$ Chances are not too high that the search will make significant progress.

Example VI: Partial Consideration of Constraints

- Sometimes the number of side constraints is way too large to consider all constraints simultaneously.
- In some modeling approaches, the number of side constraints may even be infinite.

  $\longrightarrow$ Cf. Slides nos. 20 ff.
- In cases like these, one can try to approximate the optimal solutions by (typically infeasible) "solutions":
  - Every finite subset of the set of all side constraints constitutes a certain (potentially infeasible) "solution".
  - Try to find one of these "solutions" S such that S is acceptably close to at least one of the actual optimal solutions.

Partial consideration of constraints (cont'd):

- For an instance $I \in \mathcal{I}$ let $\mathcal{C}(I)$ denote the set of constraints.
- For each finite set $C \subseteq \mathcal{C}(I)$ of constraints, let S_C denote some arbitrary yet fixed optimal solution to C.
- *Intent*:
  - Suppose we have a solver for the case that the number of constraints is finite.
  - For $C \in \mathcal{C}(I)$, S_C is then the solution delivered by this solver.
- This constitutes an alternate set

$$F^*(I) = \{S_C : C \subseteq \mathcal{C}(I); \#C < \infty\}$$

  of "feasible" solutions.

Note: $F^*(I) \cap F(I) = \emptyset$ is possible.

Example:

- *Input*: $n \in \mathbb{N}$, $c, w \in \mathbb{R}^n$, $r \in \mathbb{R}$.
- *Ground set* for the feasible outputs: $\mathbb{R}^n$
- *Side constraints*: for each $v \in \mathbb{Q}^n$ such that $\|v\|_2 = 1$, a side constraint $v^T(x - c) \leq r$.

  $\longrightarrow$ Describes a ball of radius r around c.
- *Objective*: maximizing $w^T x$ (a linear function).
- A finite selection of side constraints amounts to a finite selection $v_1, \ldots, v_k$ of vectors v:
$$\forall i \in \{1, \ldots, k\} : v_i^T(x - c) \leq r\,.$$
- For any finite selection $v_1, \ldots, v_k$, S_C is unique.
- If $w \notin \{v_1, \ldots, v_k\}$, S_C is not in this ball.

  $\longrightarrow$ The last two statements left as an easy exercise in linear algebra.

Polynom approximation revisited:

- Cf. Slides nos. 22 ff.
- A finite selection of side constraints amounts to a finite selection of values $x_1, \ldots, x_n \in [a \ldots b]$.
- Resulting finite set of side constraints:

$$\forall j \in \{1, \ldots, n\} : \left| f(x_j) - \sum_{i=0}^{k} u[i] \cdot x_j^i \right| \leq z \,.$$

Neighborhoods for selections of side constraints:

- On finite selections of side constraints, one can easily define various neighborhood relations.

- *Simple example*:

  Two sets of side constraints are neighbored if one set is constructed from the other one by inserting, removing, or exchanging exactly one side constraint.

- Example *polynom approximation*:
  "exchanging" means that one x_i is moved to another position inside $[a \dots b]$.

Section 2.2: Exact Neighborhoods

Exact Neighborhoods

- From Slide no. 47, recall that the primitive local–search scheme delivers a global optimum in case of an exact neighborhood.

- Here, we will consider a few examples of exact neighborhoods.

Example VII: Convex Programming

- **Recall the problem definition:** Convex programming means the restriction of the general optimization problem to minimization problems in which
  - each $F(I)$, $I \in \mathcal{I}$, is a convex subset of some space $\mathbb{R}^n$ and
  - the objective function to be minimized is also convex.
- For each positive integer n, let
  - $d_n : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}_0^+$ be some distance function on the space $\mathbb{R}^n$ (e.g. the Euclidean distance) and
  - $r_n > 0$ be some positive real number.
- Given an instance I with $F(I) \subseteq \mathbb{R}^n$, we specify for all $s_1, s_2 \in F(I)$:

$$(s_1, s_2) \in A \;:\Longleftrightarrow\; d(s_1, s_2) \leq r_n \,.$$

- *Claim*: This neighborhood relation is exact.

  $\longrightarrow$ Proof left as an easy exercise in basic calculus.

Example VII: Matching

Problem definition:

- Recall this problem from Slides nos. 26 ff.
- *Details reviewed*:
  - *Input*: an undirected graph $G = (V, E)$.
  - *Feasible output*: a set $M \subseteq E$ such that no two edges in E have an incident node in common.

    $\longleftrightarrow$ M is called a *matching* of G.
  - *Objective*: maximizing $\#M$.
- In the following, M denotes an arbitrary but fixed matching in some graph $G = (V, E)$.

Alternating Paths

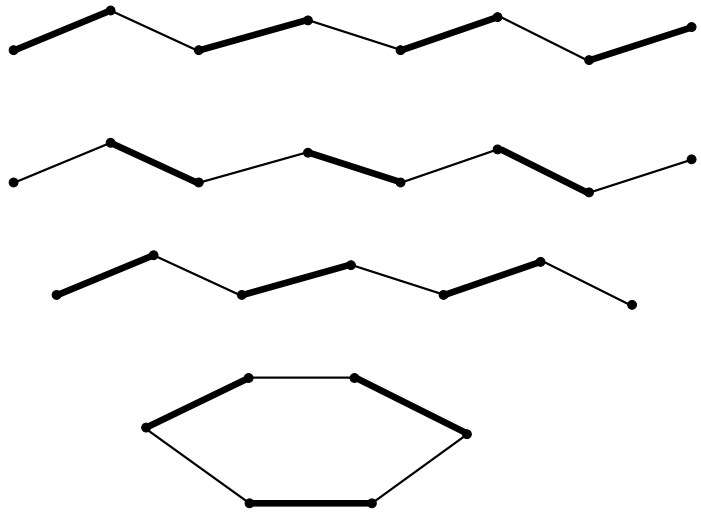
- An elementary path p in G is called *alternating* (with respect to M) if exactly every second edge of p belongs to M.

  $\longrightarrow$ In other words, the edges of M appear on p in an alternating fashion.

- *More specifically*: Let $e_1 - e_2 - e - \cdots - e_{k-1} - e_k$ be the edges on p in the order in which they appear on p.
  - Either we have $e_i \in M$ for all odd $i \in \{1, \ldots, k\}$ and $e_i \notin M$ for all even $i \in \{1, \ldots, k\}$.
  - Or we have $e_i \in M$ for all even $i \in \{1, \ldots, k\}$ and $e_i \notin M$ for all odd $i \in \{1, \ldots, k\}$.

- This definition includes elementary cycles.

Examples of alternating paths:



$\longrightarrow$ Obviously, there are four fundamentally different shapes of alternating paths, which are represented by these four small examples.

Neighborhood Relation for Matching

Two matchings M_1 and M_2 in an undirected graph $G = (V, E)$ are neighbored if the symmetric difference

$$M_1 \triangle M_2 = (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$$

is a single alternating path of both M_1 and M_2.

Claim: This neighborhood relation is exact.

Proof: On the very next slide.

Remarks:

- The matching problem is a *maximization* problem, and the objective value $c(s)$ of a matching s is the number of edges in s.
- Clearly, in the loop on Slide no. 47 this means that $s^* \triangle s$ is an alternating path, which is (w.r.t. s^*!) of the second kind from the last slide (paths of odd length and both endnodes are exposed).
- The latter paths are called *M–augmenting paths*.
- Reformulation of a famous **Theorem of Berge (1957)**: Let G be a graph with some matching M. Then M is maximum if and only if there is no M-augmenting path.

Proof of the claim from the last slide:

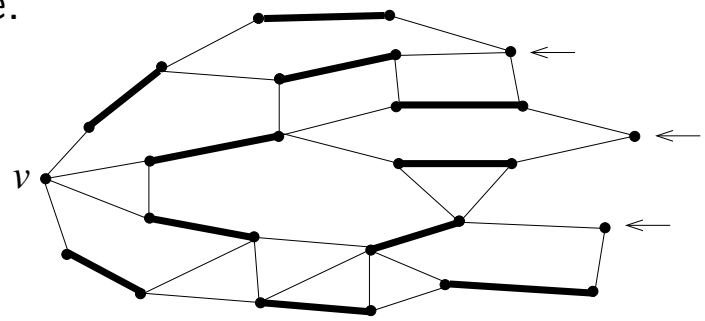
- For convenience, we will identify a path with its set of edges.
- Let M_1 and M_2 be two matchings such that $\#M_2 > \#M_1$.
- *We have to show*: In this case there is an alternating path p for M_1 such that $\#(M_1 \triangle p) > \#M_1$.
- Clearly, at most one edge of each of M_1 and M_2 is incident to a given node.

  $\longrightarrow$ Every node has degree at most two in the symmetric difference $M_1 \triangle M_2$.

  $\longrightarrow$ $M_1 \triangle M_2$ decomposes into isolated paths and cycles, which are all alternating for both M_1 and M_2.
- Since $\#M_2 > \#M_1$, at least one of these paths p, say, must contain more arcs from M_2 than from M_1.
- Clearly, this is a path p as desired.

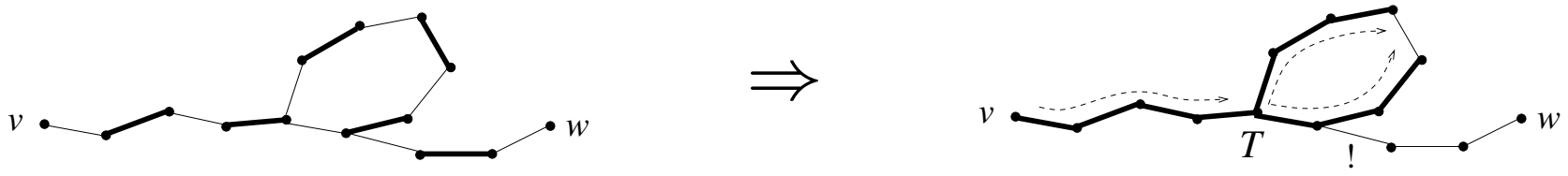
More details of local search for matching:

- A start solution for the local search scheme on Slide is easy to find: the empty matching.
- At first glance, it also seems easy to find an alternating path:
  - Find an unmatched node $v \in V$, that is, a node that is not incident to any edge of the current matching s^*.
  - Determine all nodes $w \in V$ such that there is an alternating (v, w)–path.
  - If at least one of these nodes w is unmatched, the symmetric difference of the current matching and this (v, w)–path is a matching with one more edge.



Searching for Alternating Paths

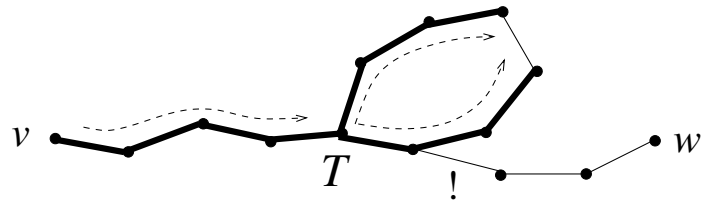
- Clearly, we cannot enumerate all possible alternating paths that start with v, because their number may be exponentially large ($\longrightarrow$ left as an exercise).
- Each of the common efficient search strategies (depth–first, breadth–first, ...) determines a tree T, which clearly contains at most one (v, w)–path for each node w.
- However, it can be seen (formal details omitted; see the picture below for an intuition) that, possibly, alternating paths to some nodes w (and thus the nodes themselves) may be missed.



Searching Alternating Paths (cont.)

- Embarrassingly, this gap has not been closed for about ten years (from the mid–fifties to the mid–sixties), until Jack Edmonds had a brilliant insight.
- Fortunately, there is an efficient (yet complicated) "workaround".
- *Precise problem*: cycles of odd length such that
  - a maximal number of edges on this cycle belong to the matching,
  - the remaining node is also matched, and
  - tree T enters the cycle via this node.

  $\longrightarrow$ Called a *blossom* in the literature.



Blossom Shrinking

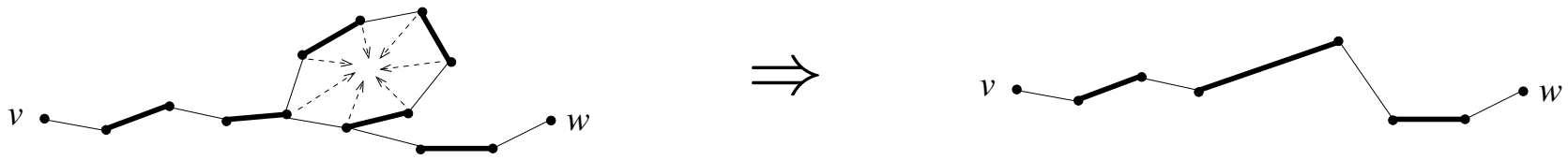
Crucial insight (proof omitted):

If we "shrink" the blossom to a single node, then:

There is an augmenting (v, w)–path in the original instance equipped with the original matching.

$\Longleftrightarrow$

There is an augmenting (v, w)–path in the reduced instance equipped with the correspondingly reduced matching.



Edmonds's Matching Algorithm (Sketch)

- Since we cannot determine a better neighbored solution in a reasonable way, we cannot apply the "pure" local–search scheme.
- However, we can modify the loop on Slide no. 47:
  - We search for an alternating path by growing a set of alternating trees (alternating forest).
  - If we detect a blossom, we shrink this blossom into a pseudonode and continue with the resulting graph and matching.
  - If we find an alternating path p connecting two unmatched nodes, we replace the current matching M by $M \triangle p$, expand pseudonodes on this path (recursively), and continue as in the regular local–search scheme.
  - Finally, if neither case applies, we have found a maximal matching in the shrunken graph (proof omitted).
- At the very end, all shrinking operations are undone and the matching is extended to all blossom edges.

Example VIII: Minimum Cost Flows

- *Input*:
  - a directed graph $D = (V, A)$;
  - *lower and upper capacity values* $0 \leq \ell[a] \leq u[a] \in \mathbb{R}$ and a *cost factor* $c[a] \in \mathbb{R}$ for each arc $a \in A$;
  - a *balance value* $b[v] \in \mathbb{R}$ for each node $v \in V$.
- *Desired output*: a *flow value* $f[a] \in \mathbb{R}$ for each arc $a \in A$ such that
  - $\ell[a] \leq f[a] \leq u[a]$ for each arc $a \in A$ *(capacity constraints)* and
  - for each node $v \in V$:

$$\sum_{\substack{w \in V \\ (v,w) \in A}} f[(v,w)] \; - \sum_{\substack{w \in V \\ (w,v) \in A}} f[(w,v)] \; = \; b[v] \, .$$

    (flow balance constraints)
- *Objective*: minimizing $\sum_{a \in A} c[a] \cdot f[a] \, .$
- If $b \equiv 0$, flows are called *circulations*.

Neighborhood relation for Min-Cost-Flows:

- In the following, let (G, ℓ, u, b), $G = (V, A)$, be an arbitrary, yet fixed instance.
- *Neighborhood relation*:
  - Let f_1 and f_2 be two feasible flows for (G, ℓ, u, b),
  - Then f_1 and f_2 are called neighbored if they only differ on one elementary weak cycle.

Claim: This neighborhood relation is exact.

Proof: On the next few slides (until Slide no. 84).

Remark:

The application of the algorithmic scheme from Slide no. 47 to the min–cost flow problem on this neighborhood relation is called the *negative–cycle canceling algorithm* in the literature.

Terminology:

- A path is *elementary* (*simple*) if it contains each of its nodes only once.
- In the following, paths and cycles
  - are defined as sets of (directed) arcs
  - and may contain arcs in forward and backward direction ($\longrightarrow$ weak paths or cycles, respectively).
- Let p be such a weak path in a directed graph $G = (V, A)$.

Augmenting paths:

- Let $A_{\text{forw}}(p)$ and $A_{\text{backw}}(p)$ denote the forward and backward arcs of p, respectively.
- Then p is called *augmenting* w.r.t. an arc weighting $f : A \longrightarrow \mathbb{R}$, lower bounds $\ell : A \longrightarrow \mathbb{R}$ and upper bounds $u : A \longrightarrow \mathbb{R}$ if
  - $f[a] < u[a]$ for all $a \in A_{\text{forw}}(p)$ and
  - $f[a] > \ell[a]$ for all $a \in A_{\text{backw}}(p)$.
- Two weak paths p_1 and p_2 are *consistent* if
  - $A_{\text{forw}}(p_1) \cap A_{\text{backw}}(p_2) = \emptyset$ and
  - $A_{\text{backw}}(p_1) \cap A_{\text{forw}}(p_2) = \emptyset$.

Negative Cycles

Terminology (cont'd):

- For a weak cycle p and $\varepsilon > 0$, $\varepsilon \cdot p$ denotes the circulation $f : A \longrightarrow \mathbb{R}$ defined by
  - $f(a) = +\varepsilon$ for $a \in A_{\mathsf{forw}}(p)$,
  - $f(a) = -\varepsilon$ for $a \in A_{\mathsf{backw}}(p)$,
  - $f(a) = \quad 0$ for $a \in A \setminus p$.
- A cycle is called *negative* w.r.t. some cost function $c : A \longrightarrow \mathbb{R}$ if

$$\sum_{a \in A_{\mathsf{forw}}(p)} c\,[a] \;-\; \sum_{a \in A_{\mathsf{backw}}(p)} c\,[a] \;<\; 0\,.$$

  $\longrightarrow$ If, and only if, $\varepsilon \cdot p$ has negative cost w.r.t. c for $\varepsilon > 0$.

Flow Decomposition

Lemma:

For two feasible flows f_1 and f_2 for (G, ℓ, u, b), there are some non-negative integer k and weak cycles $p_1, \ldots, p_k$ such that:

- There are values $\varepsilon_1, \ldots, \varepsilon_k > 0$ such that

$$f_2 = f_1 + \varepsilon_1 \cdot p_1 + \cdots + \varepsilon_k \cdot p_k \,.$$

- For each cycle p_i, $i \in \{1, \ldots, k\}$, we have
  - $f_2[a] > f_1[a]$ for all $a \in A_{\mathsf{forw}}(p_i)$ and
  - $f_2[a] < f_1[a]$ for all $a \in A_{\mathsf{backw}}(p_i)$.

  $\longrightarrow$ In particular, $p_1, \ldots, p_k$ are pairwise consistent.

Proof: By induction on the number of arcs with $f_1[a] \neq f_2[a]$. Details are left as an exercise.

Exactness of the Min-Cost-Flow Neighborhood

Proof:

- Suppose that f_1 is not optimal.

  $\longrightarrow$ To prove the claim, it then suffices to show that there is some negative cycle p that is augmenting w.r.t. f_1, the lower bounds ℓ, and the upper bounds u.

- Suppose that f_2 is optimal.

  $\longrightarrow$ The cost of f_2 is strictly smaller than the cost of f_1.

  $\longrightarrow$ Among the cycles $p_1, \ldots, p_k$ that are guaranteed by the flow decomposition lemma, at least one must be negative.

- Let p_i denote this cycle.

- Since p_i is augmenting, $f_1 + \varepsilon_i \cdot p_i$ is obviously feasible.

- In summary, p_i is a cycle as desired.

Concluding Remarks on Exact Neighborhood Relations

- If a neighborhood relation is not exact, there is still some (heuristic!) hope that the solution from the algorithmic scheme on Slide no. 47 is not "too bad".
- However, the search will always be "trapped" in a local optimum "near" the start solution.
- Such a local optimum may be very bad compared to the overall ("global") optimum.
- In the following, we will discuss a couple of heuristic techniques to let the search "escape" from local optima.