Facility location problems

Discrete models and local search methods

Yuri Kochetov

Sobolev Institute of Mathematics
Novosibirsk Russia
e-mail: jkochet@math.nsc.ru

Lecture 3

Computational Complexity of Local Search

Content

1. The class PLS
2. PLS–reductions and PLS–completeness
3. A first PLS–complete problem
4. PLS complete facility location problems
5. Complexity of the standard local search algorithm

Definition 3.1. An optimization problem OP is characterized by the following *quadruple of objects* (I, Sol, F, $goal$), where

- I is the set of instances of OP;
- Sol is a function that associates to any input instance $x \in I$ the set of feasible solutions of x;
- F is the measure function, defined for pairs (x, s) such that $x \in I$ and $s \in Sol(x)$. For every such pair (x, s), $F(s)$ provides a positive integer which is the value of the feasible solution s;
- $goal \in \{\min; \max\}$ specifies whether OP is a maximization or a minimization problem.

We want to find global optimal solution

Definition 3.2. A local search problem is defined by the pair $L = (OP, N)$, where OP is optimization problem and N: $Sol(\mathrm{x}) \to 2^{Sol(x)}$ is a neighborhood function. The $N(s, x)$ is called the *neighborhood* of the solution $s \in Sol(x)$. For given an instance $x \in I$, we want to find a locally optimal solution.

Let $L_1 = (OP, N_1)$, $L_2 = (OP, N_2)$ are two local search problems. We say that neighborhood N_2 *stranger* than neighborhood N_1 ($N_1 \preccurlyeq N_2$) if each local optimum for N_2 neighborhood is local optimum for N_2 neighborhood.

The class PLS

We assume that instances and solutions are encoded as binary strings, and $|s| \leq p(|x|)$ for each $s \in Sol(x)$.

Definition 3.3. A local search problem L is *in PLS* if there are three polynomial–time algorithms A_L, B_L, C_L with the following properties:

– Given a string $x \in \{0, 1\}^*$, algorithm A_L determines whether x is an instance x∈I, and in this case it produces some solution $s_0 \in Sol(x)$.

– Given an instance x and a string s, algorithm B_L determines whether $s \in Sol(x)$ and if so, B_L computes the cost $F(s, x)$ of the solution s.

– Given an instance x and a solution s, algorithm C_L determines whether is a local optimum, and if it is not, C_L outputs a neighbor $s' \in N(s, x)$ with (strictly) better cost, i.e., $F(s, x)$ for minimization problem, and $F(s', x) > F(s, x)$ for maximization problem.

Theorem 3.1. [Johnson, Papadimitriou, Yannakakis] If a PLS problem L is NP–hard, then NP = co–NP.

Proof. If L is NP–hard, there is NP–complete decision problem D which can be solved by polynomial deterministic algorithm M with an oracle. This oracle solves the local search problem L and returns a local optimum. Running time of the oracle is ignored.

Let us consider the complementary decision problem D^c. If $D^c \in$NP then NP = co–NP (see M. Garey, D. Johnson, Computers and Intractability, Theorem 7.2). To show $D^c \in$NP we need a polynomial nondeterministic algorithm for D^c. Let algorithm M' repeats computations of M and guesses a local optimum for L instead of to call of the oracle. At the final step, M' returns «yes», if M returns «no». Notice that M' is polynomial because M is polynomial and algorithm C for L (definition 3) can check local optimality of the guess in polynomial time. So $D^c \in$NP. ■

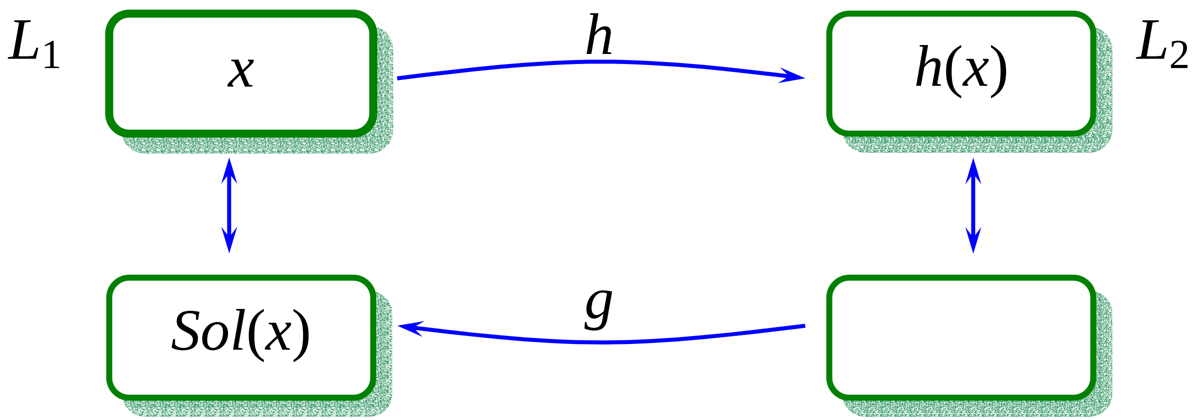
PLS–reductions

Definition 3.4. Let L_1 and L_2 be two local search problems. A *PLS–reduction* from L_1 to L_2 consists of two polynomial time computable functions h and g such that

a) h maps instances x of L_1 to instances $h(x)$ of L_2,

b) g maps (solution of $h(x)$, x) pairs to solutions of x,

c) for all instances x of L_1, is s is a local optimum for instance $h(x)$ of L_2, then $g(s, x)$ is a local optimum for x.



Proposition 3.1. If L_1, L_2 and L_3, are problems in PLS such that L_1 PLS–reduces to L_2 and L_2 PLS–reduces to L_3, then L_1 PLS–reduces to L_1.

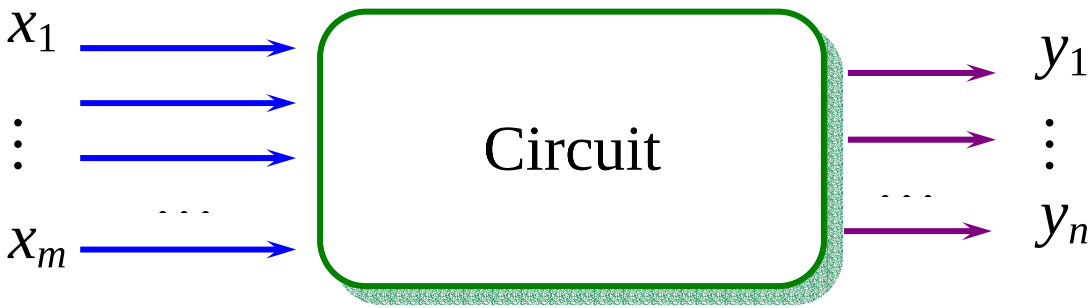
Proposition 3.2. If L_1 and L_2 are problems in PLS such that L_1 PLS–deduces to L_2 and if there is a polynomial–time algorithm for finding local optima for L_2, then there is also a polynomial–time algorithm for finding local optima for L_1.

We say that a problem L in PLS is PLS–complete if every problem in PLS can be PLS–reduced to it.

A first PLS–complete problem

(*Circuit*, *Flip*) local search problem

Input: Boolean circuit composed of $\wedge, \vee, \neg$ gates with *m* inputs and *n* output.



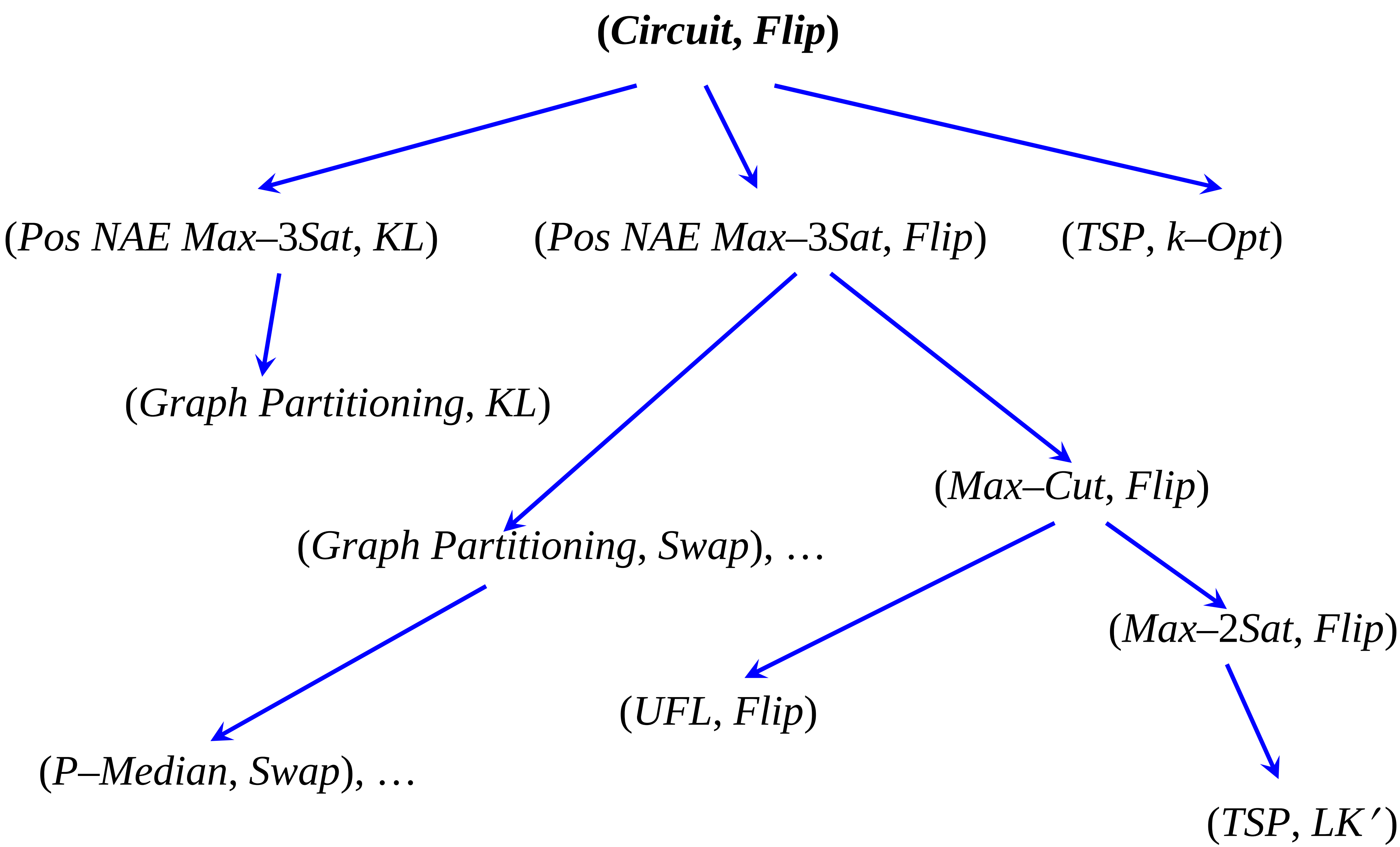
Objective function: $F(z) = \sum_{j=1}^{n} 2^{j-1} y_j(z)$

Neighborhood *Flip*(*z*) consists of all strings of length *m* that have Hamming distance 1 from *z*.

Output: String *z*.

Goal: *Flip* local minimum.

Theorem 3.2. [Johnson, Papadimitriou, Yannakakis] Both the maximization version and the minimization version of (*Circuit*, *Flip*) are PLS–complete.


(Circuit, Flip)
(Pos NAE Max–3Sat, KL)
(Pos NAE Max–3Sat, Flip)
(TSP, k–Opt)
(Graph Partitioning, KL)
(Max–Cut, Flip)
(Graph Partitioning, Swap), …
(Max–2Sat, Flip)
(UFL, Flip)
(P–Median, Swap), …
(TSP, LK′)

Theorem 3.3. The local search problem (*UFL*, *Flip*) is PLS–complete.

Proof. Let us consider the PLS–complete problem (*Max–Cut*, *Flip*). Given a graph $G = (V, E)$ with weights $w_e \geq 0$, $e \in E$.

Find a partition of the set $V = V_1 \cup V_2$ with maximal weight of the cut

$$W(V_1 V_2) = \sum (w_e \mid e = (i_1, i_2) \in E, i_1 \in V_1, i_2 \in V_2).$$

We want to reduce the problem to (*UFL*, *Flip*).

Denote by $E(i)$ the set of edges in G which are incident to the vertex $i \in V$. Put $I = V$, $J = E$ and

$$f_i = \sum_{e \in E(i)} w_e, \qquad c_{ie} = \begin{cases} 0 & \text{if } \ e = (i_1, i_2), i = i_1 \text{ or } i = i_2 \\ 2w_e, & \text{otherwise} \end{cases}.$$

For any solution $S \subseteq I$ we define a partition (V_1, V_2) by the following $V_1 = S$; $V_2 = V \setminus V_1$ and we have

$$\sum_{i \in S} f_i + \sum_{j \in J} \min_{i \in S} c_{ij} + W(V_1 V_2) = 2 \sum_{e \in E} w_e \, . \blacksquare$$

Corollary3.1. If a neighborhood N is stronger than *Flip*, then local search problem (*UFL*, N) is PLS–complete.

Theorem 3.4. The p–median problem under *Swap*, *LK*, LK_1, *FM*, FM_1 neighborhoods are PLS–complete.

Complexity of the Standard Local Search Algorithm

Definition 3.5. Let L be a local search problem and let x be an instance of L. The neighborhood graph $NG_L(x)$ of instance x is a *directed graph* with one node for each feasible solution to x, and with an arc $s \rightarrow t$ whenever $t \in N(s)$.

Definition 3.6. The *transition graph* $TG_L(x)$ is the subgraph of $NG(x)$ that includes only those arcs $s \rightarrow t$ for which the cost $F(t)$ is strictly better than $F(s)$. The height of a node v is the length of the shortest path in $TG_L(x)$ from v to a sink (a vertex with no outgoing arcs). The height of $TG_L(x)$ is the largest height of a node.

The height of a node is a lower bound on the number of iterations for the standard local search algorithm even if it uses the best possible pivoting rule.

Definition 3.7. Let L_1 and L_2 be local search problems, and let (h, g) be a PLS–reduction from L_1 to L_2. We say that the reduction is *tight* if for any instance x of L_1 the height of $TG_{L_2}(x)$ is at least as large as the height of $TG_{L_1}(x)$.

Corollary. The UFL problem under *Flip*–neighborhood is tight PLS–complete. The standard local search algorithm for this problem takes exponential number of iterations in the worst case regardless of the tie–breaking and pivoting rules used.

Corollary. The p–median problem under *Swap*, *LK*, LK_1, *FM*, FM_1 neighborhoods are tight PLS–complete. For these local search problems, the standard local search algorithm takes exponential number of iterations in the worst case regardless of the tie–breaking and pivoting rules used.

The Generalized Graph 2–Coloring Problem (2 – GGSP)

Input: Graph G = (V, E) and weights w_e, $e \in E$.

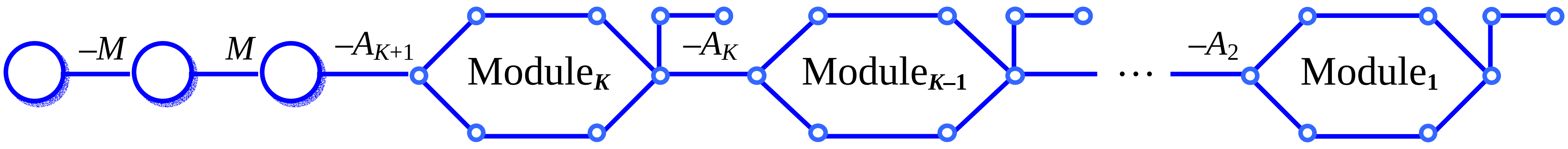
Output: A color assignment c: $V \to \{1, 2\}$

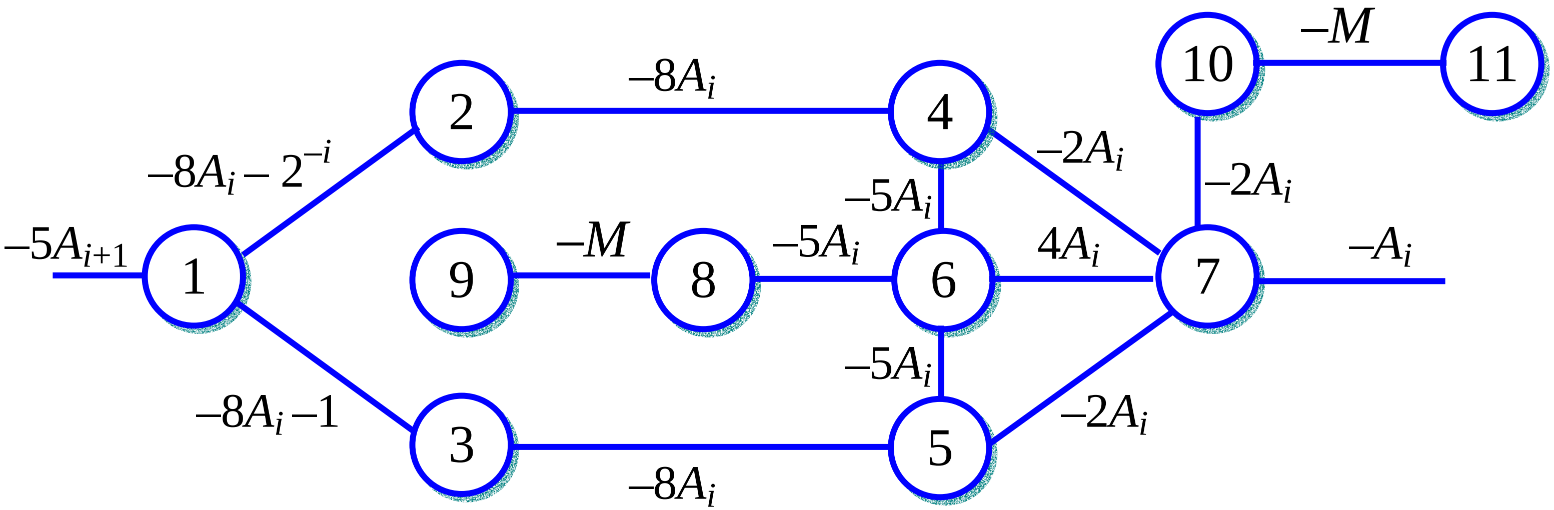
Goal: To minimize the total weight of the edges those have endpoint with the same color.

Given a solution $c(v)$, $v \in V$, a *Flip*–neighbor is obtained by choosing a node and assigning new color. A solution is *Flip*–optimal if flipping any single node does not decrease the total weight of monochromatic edges.

Theorem 3.5. [Vredeveld, Lenstra] The GGCP with the *Flip* neighborhood is tight PLS–complete.

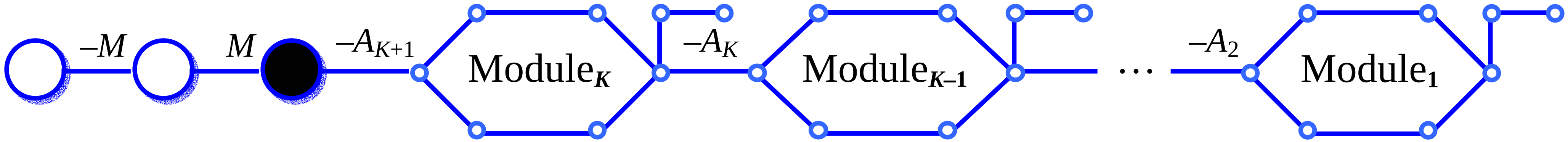
Difficult family of instances for the «Best Improvement» pivoting rule

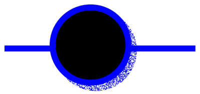


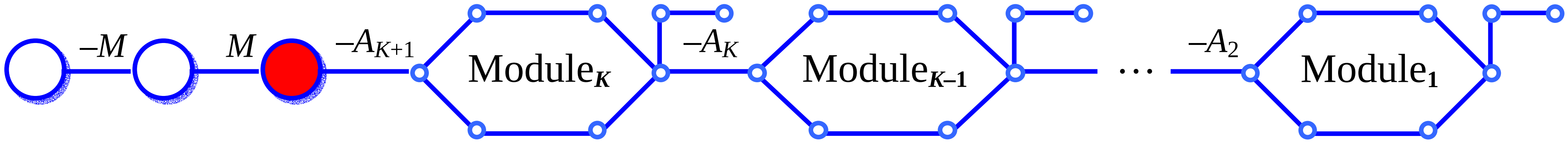


Module i : $A_i = 20^{i-1}$

Starting solution: all nodes are white. The input node of module K is only ***unhappy*** node.

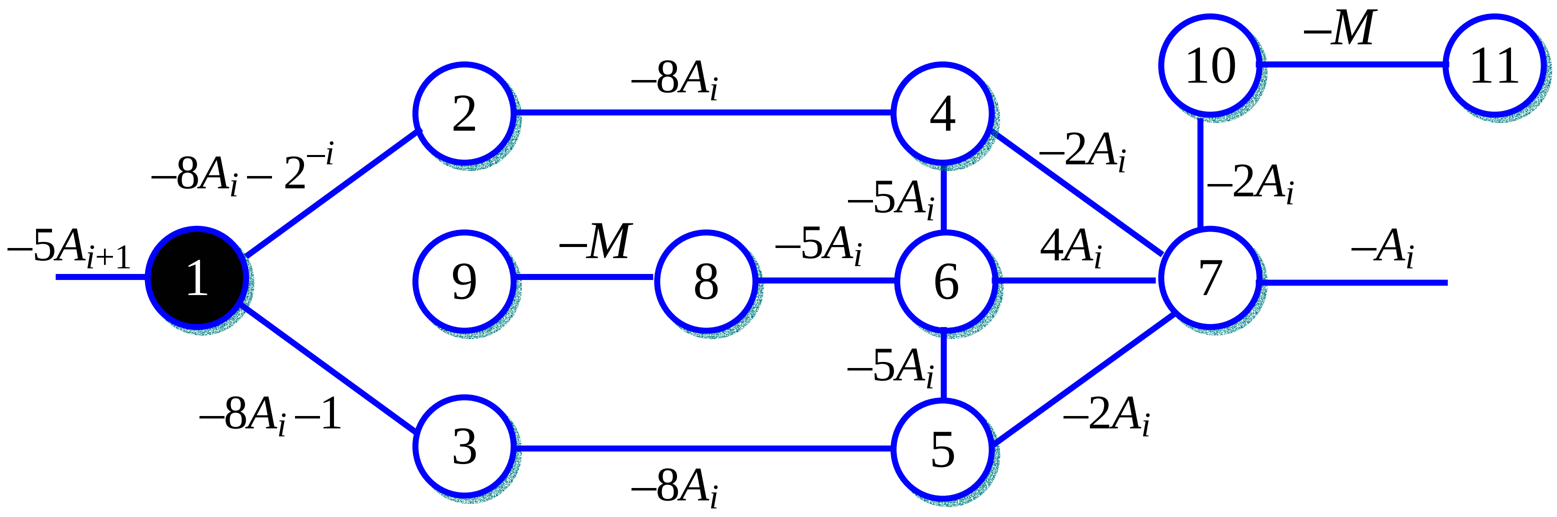


 is unhappy node. So, we are flipping this node!



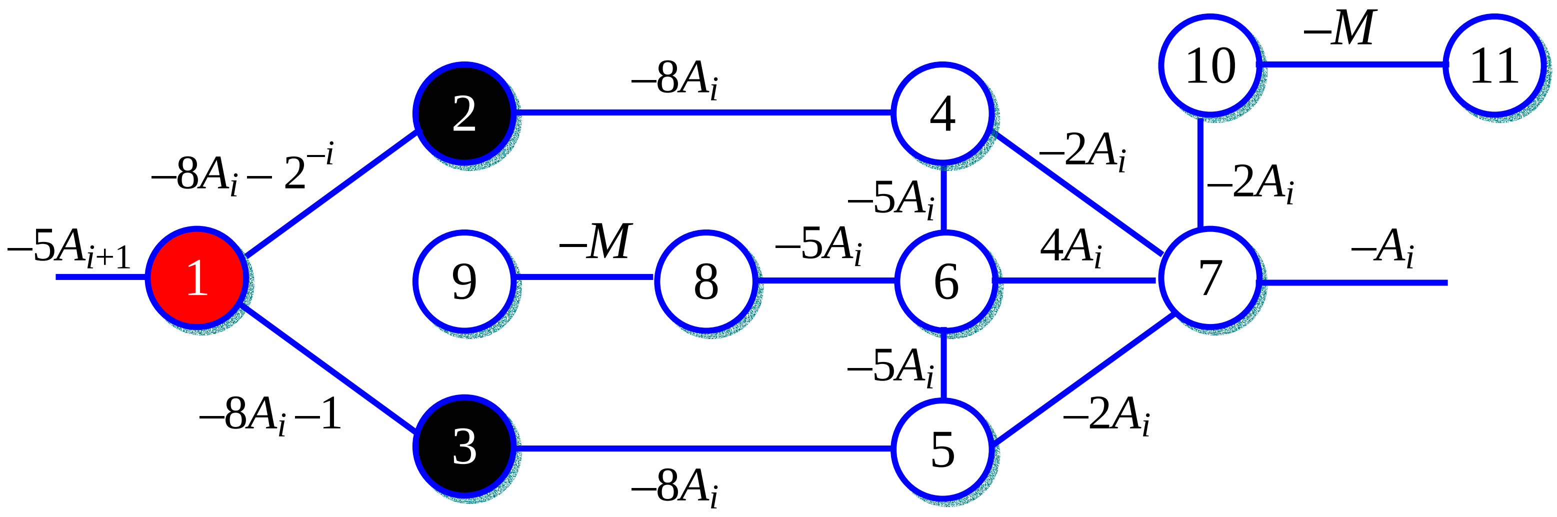
Theorem 3.6. [Vredeveld, Lenstra] If the input node of module K is the only unhappy node, the output node of module 1 flip 2^K times.

Iterations of local improvement algorithm



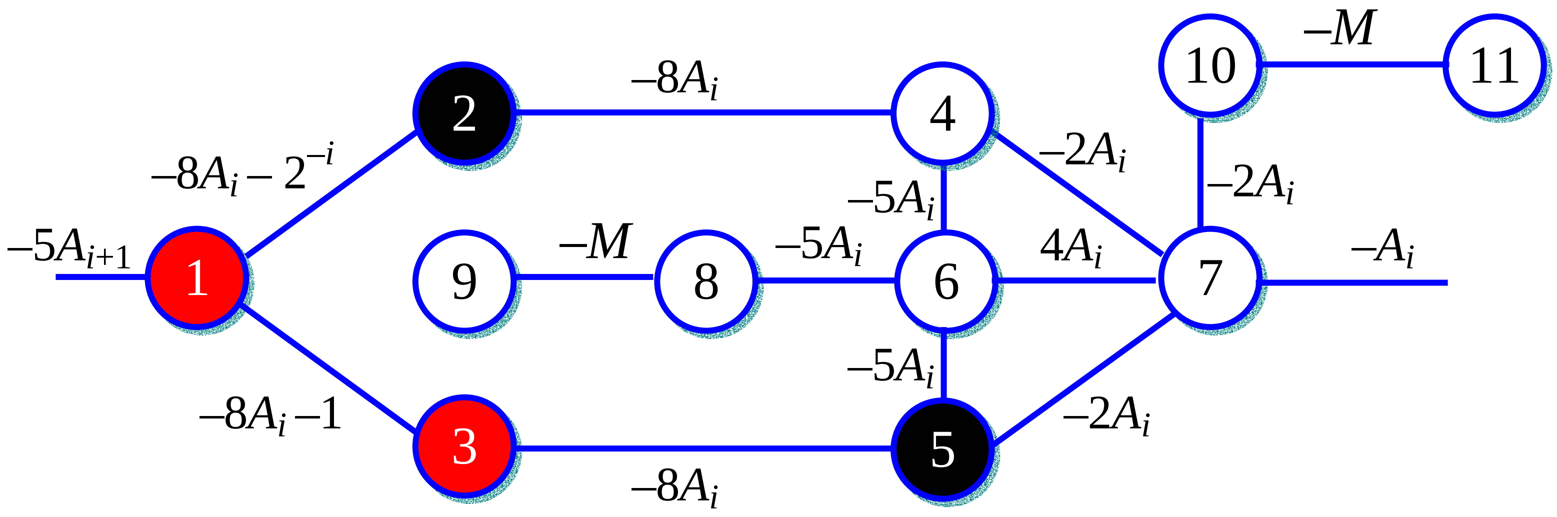
Improvement $\Delta = (8A_i + 1) + (8A_i + 2^{-i}) - 20A_i$.

Iterations of local improvement algorithm



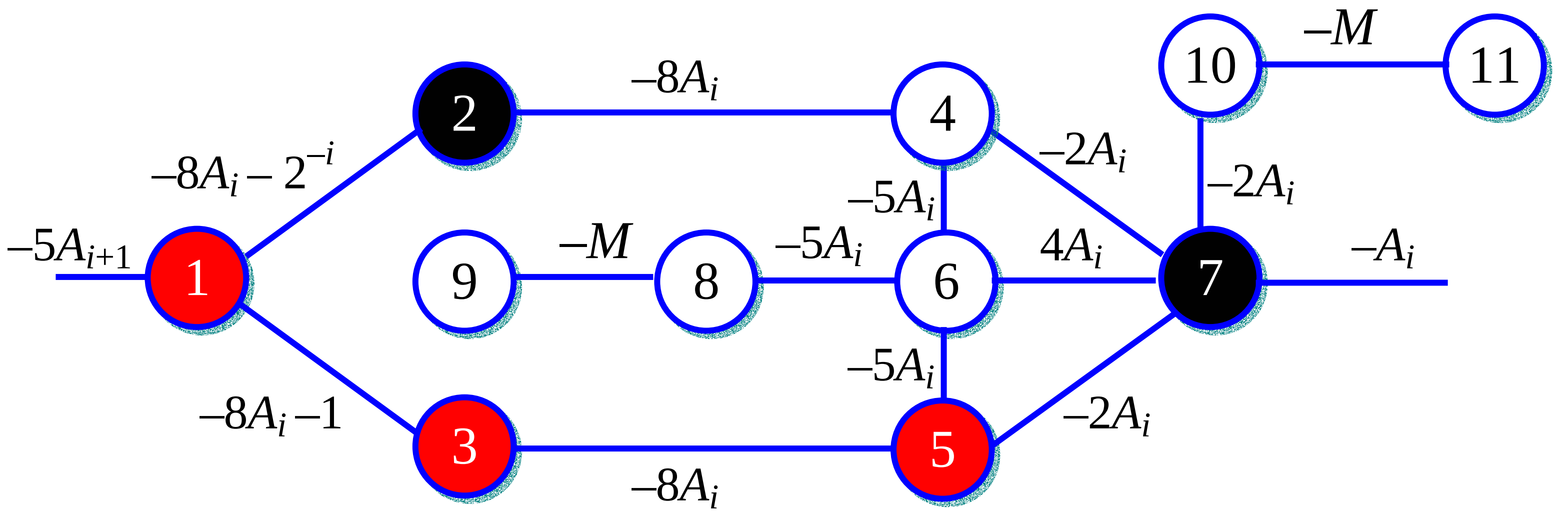
Improvement $\Delta_3 = -1,\ \Delta_2 = -2^{-i}$.

Iterations of local improvement algorithm

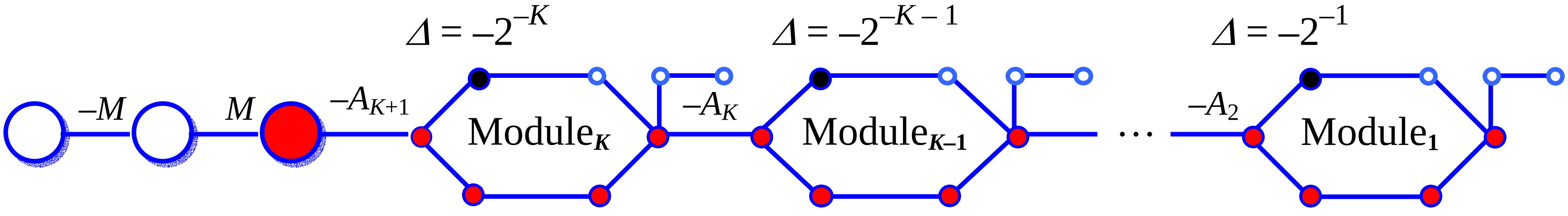


Improvement $\Delta_5 = -A_i, \;\; \Delta_2 = -2^{-i}.$

Iterations of local improvement algorithm

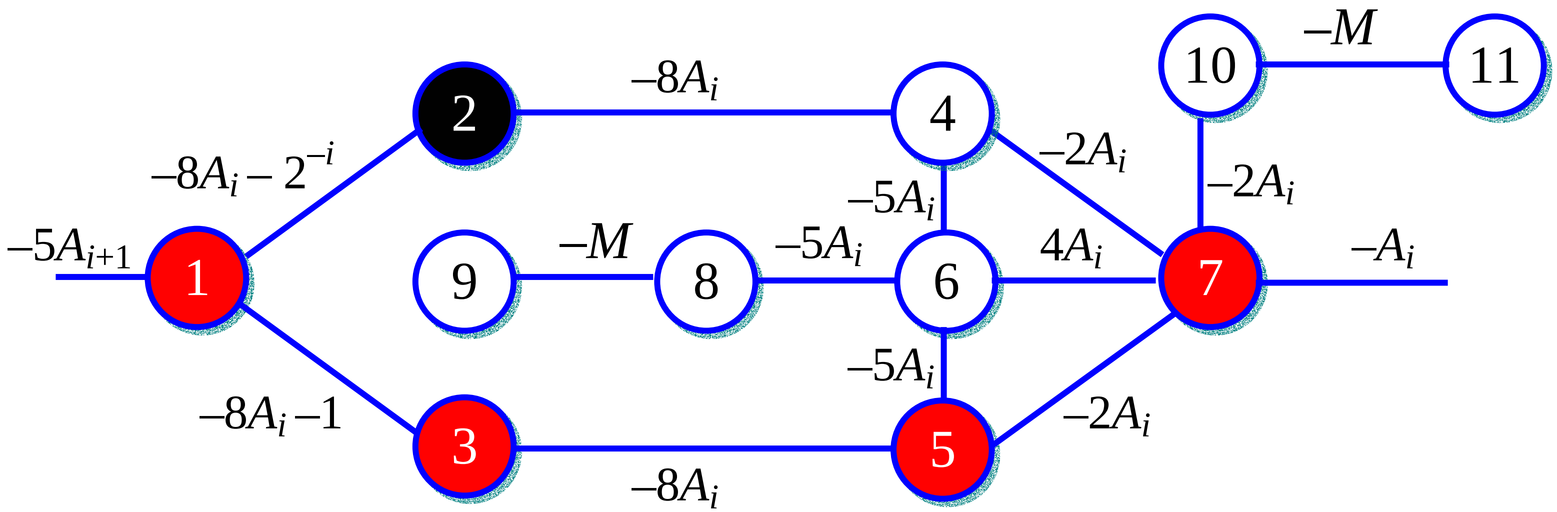


Improvement $\Delta_7 = —A_i$, $\Delta_2 = -\,2^{-i}$.



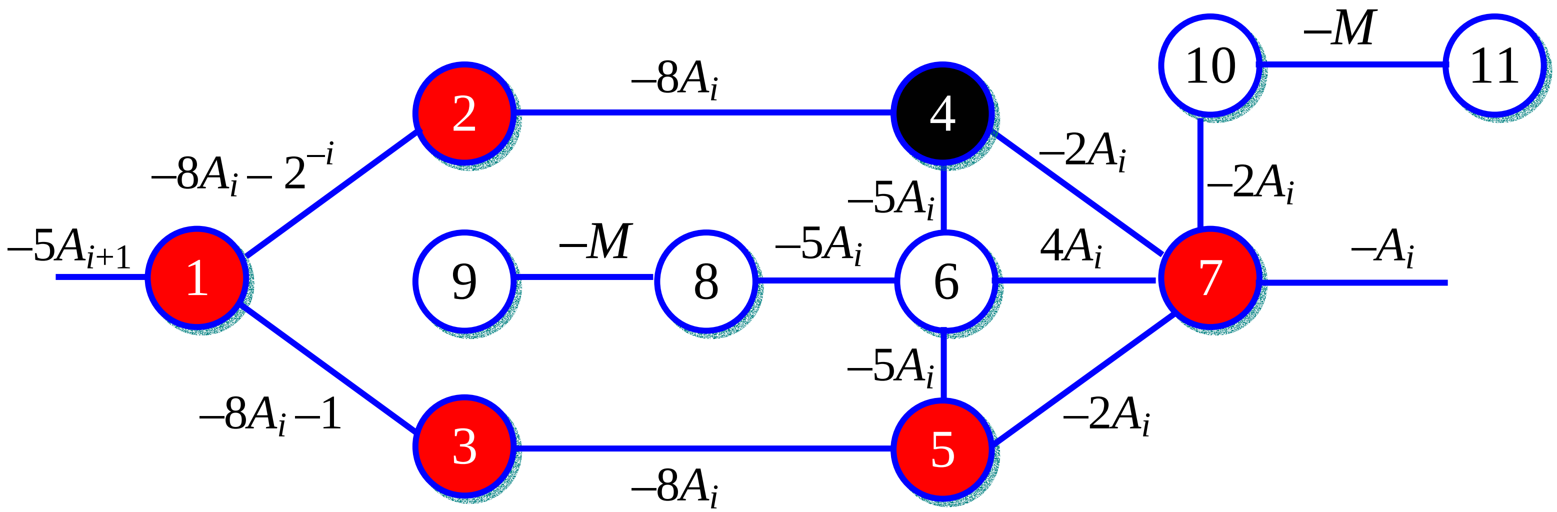
We select the best improvement $\Delta = -2^{-1}$

Iterations of local improvement algorithm



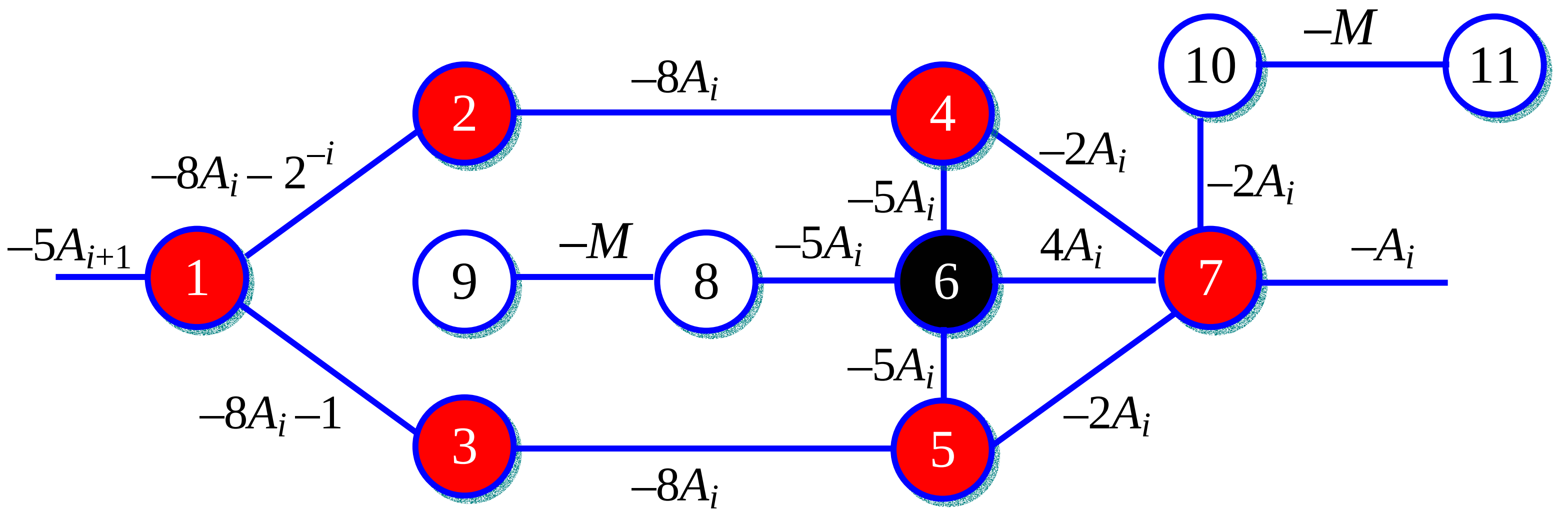
Improvement $\Delta_2 = -\,2^{-1}$.

Iterations of local improvement algorithm



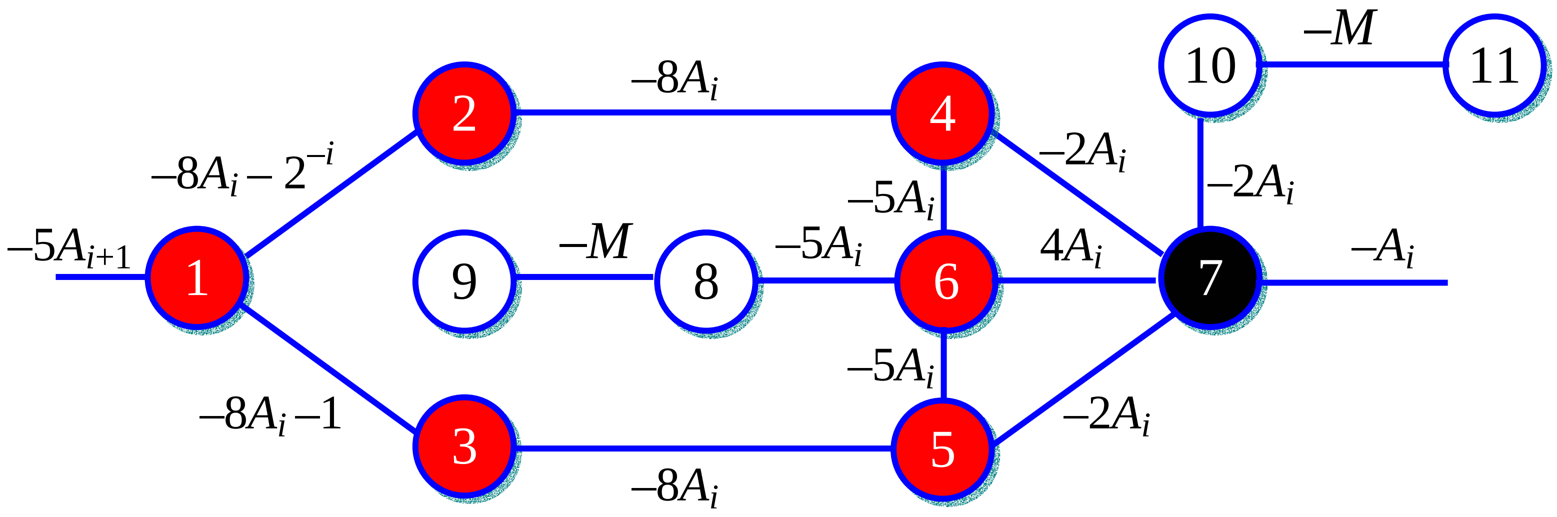
Improvement $\Delta_4 = -A_1$

Iterations of local improvement algorithm



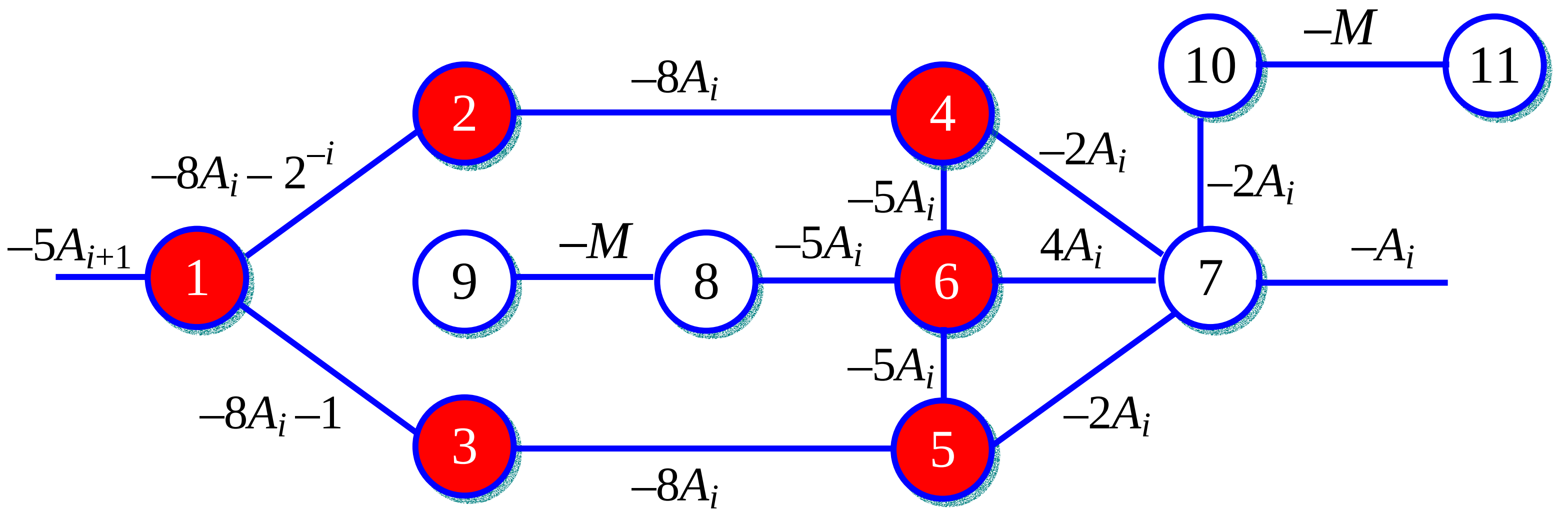
Improvement $\Delta_6 = -A_1$

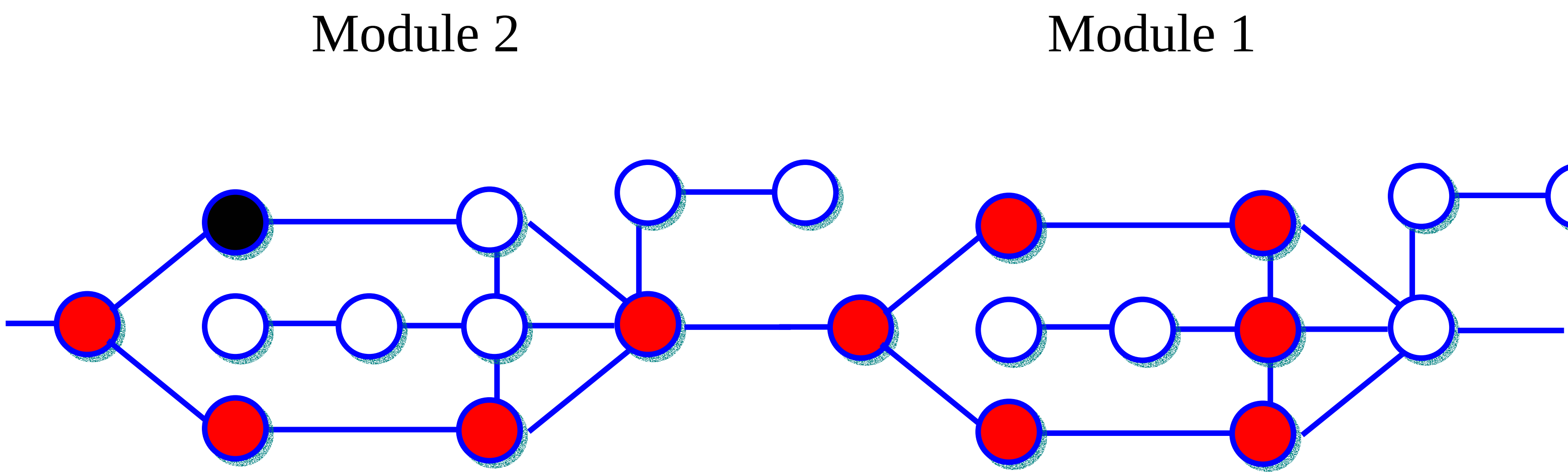
Iterations of local improvement algorithm



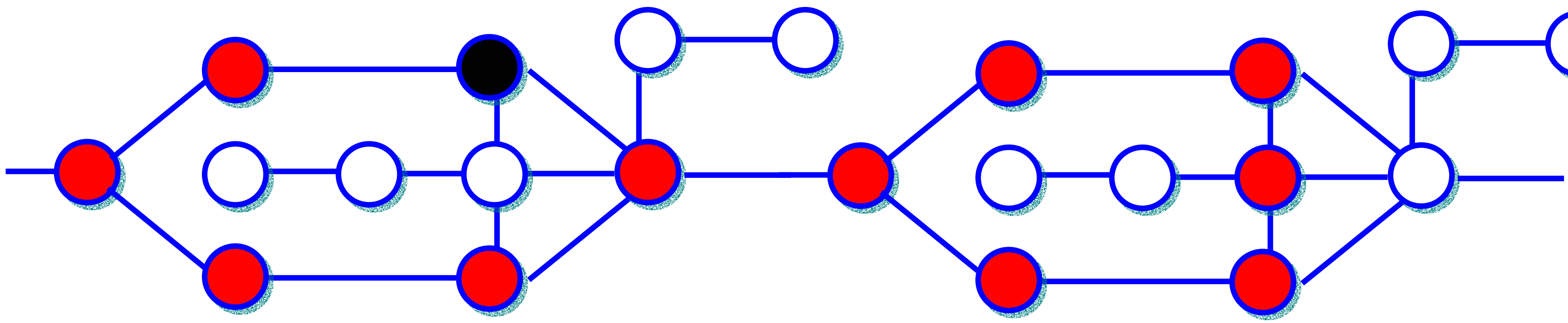
Improvement $\Delta_7 = -\,2A_1$

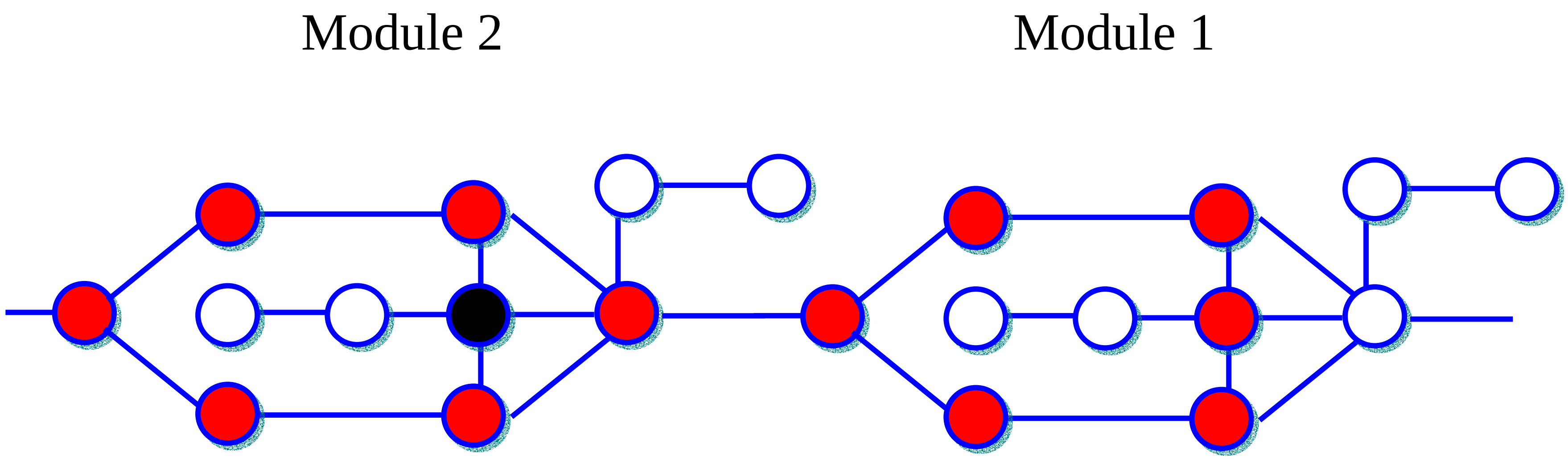
Iterations of local improvement algorithm

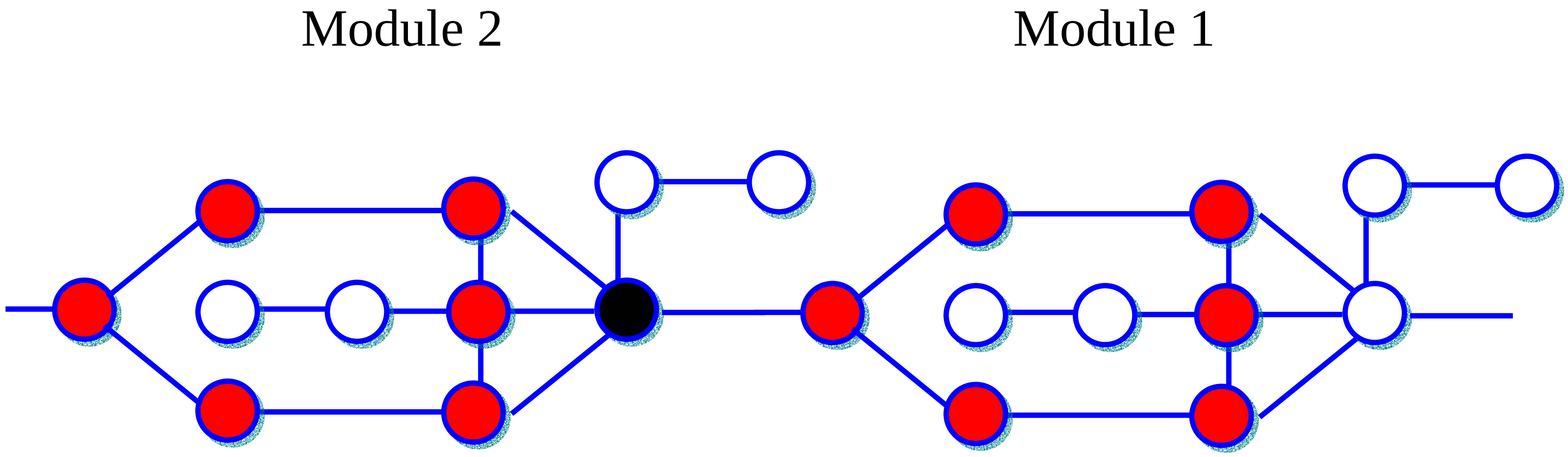


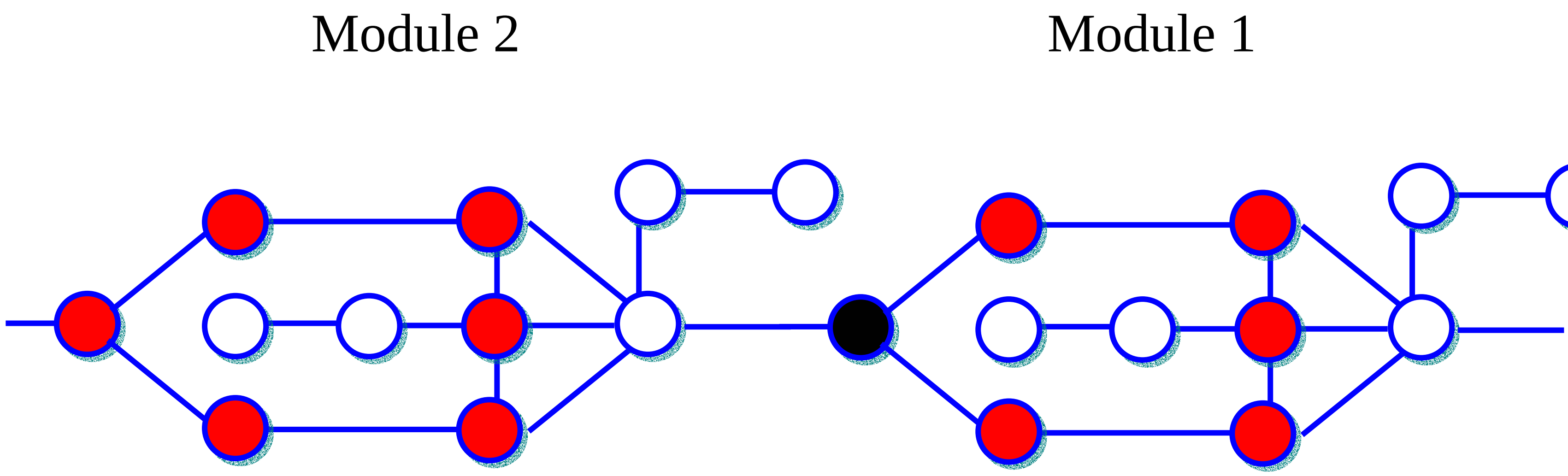

Module 2
Module 1

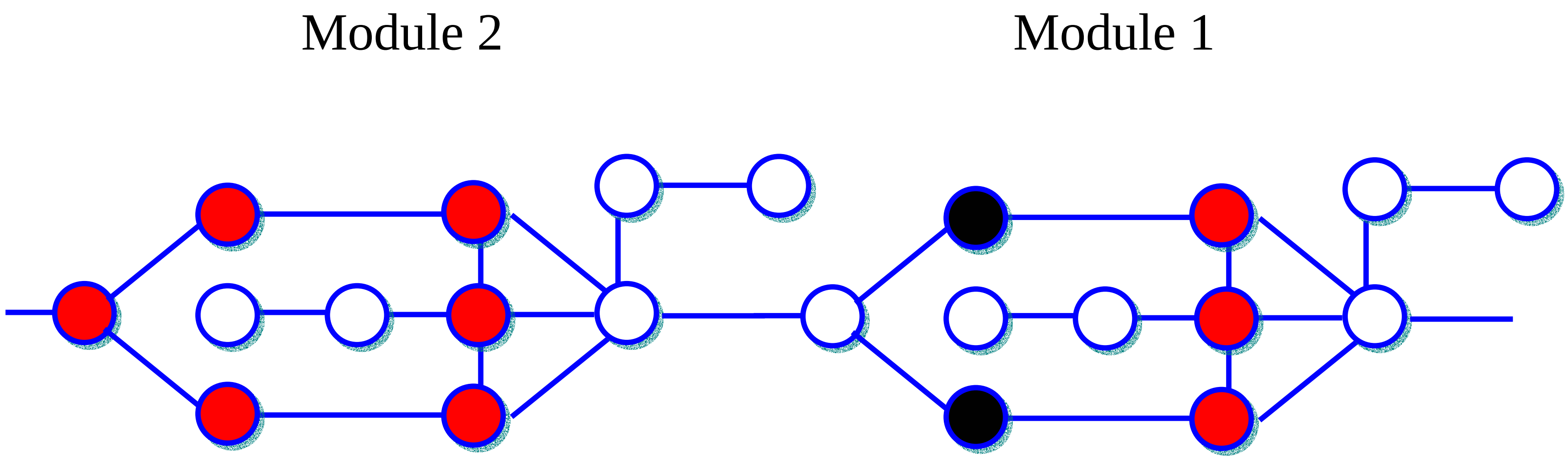
Module 2 Module 1



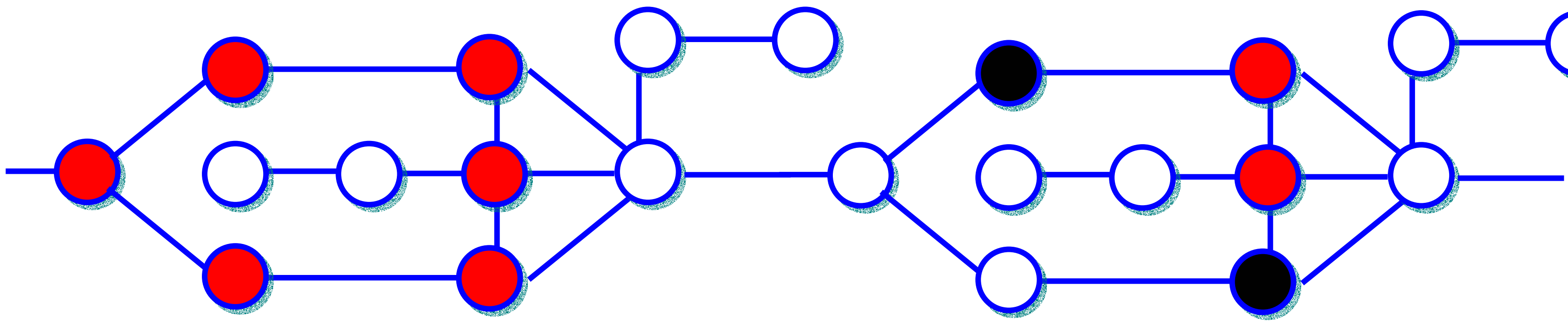

Module 2
Module 1

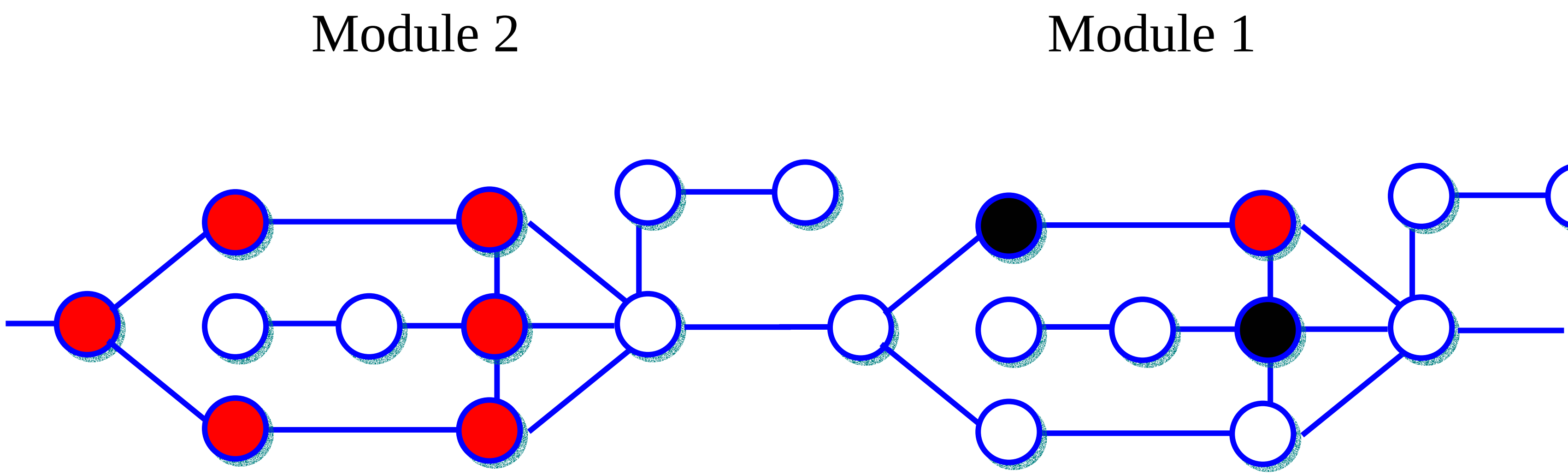

Module 2
Module 1


Module 2
Module 1

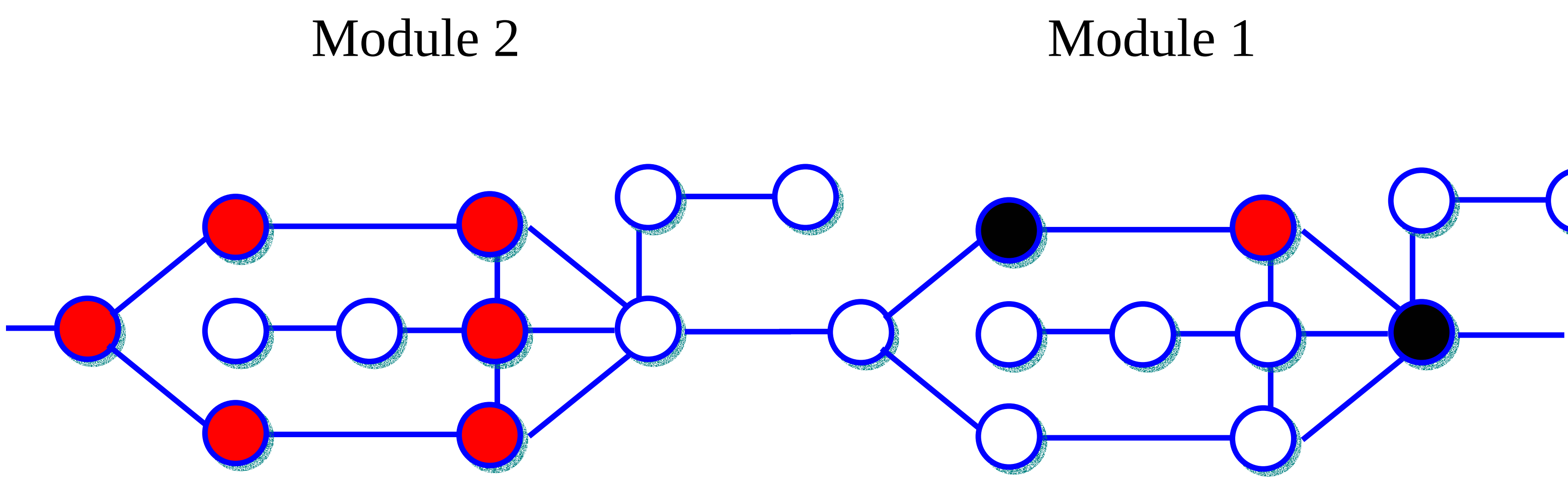

Module 2
Module 1

Module 2 Module 1

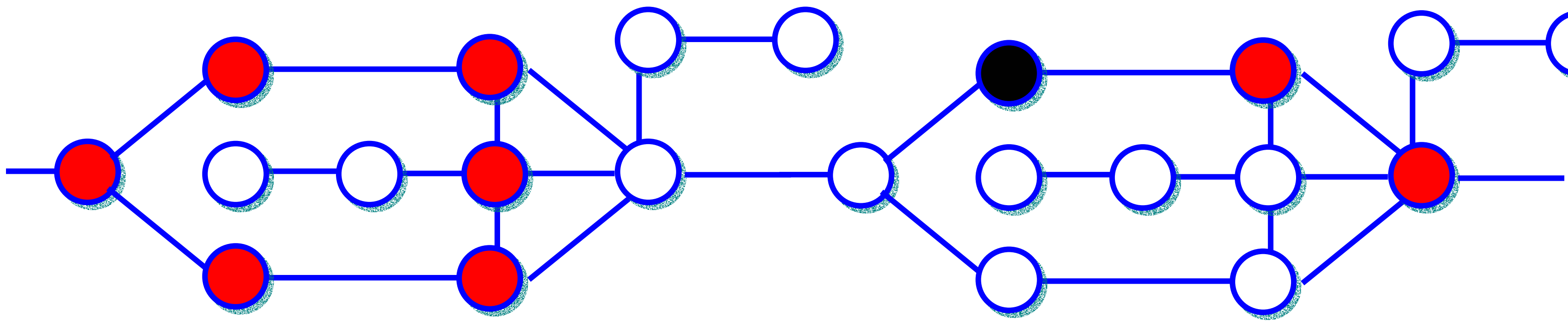


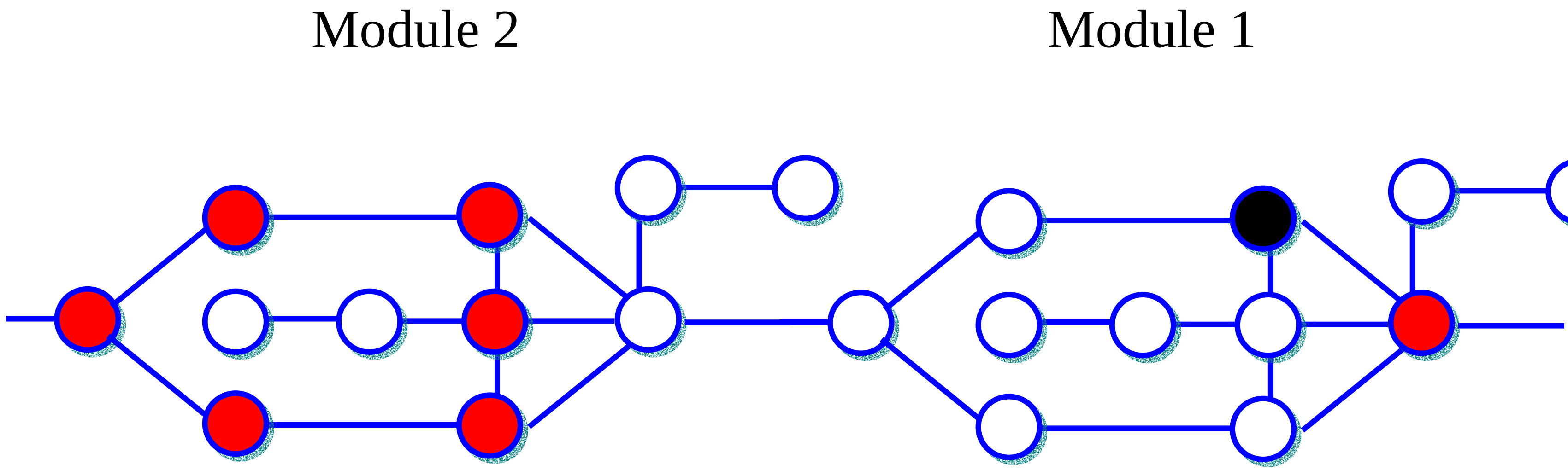


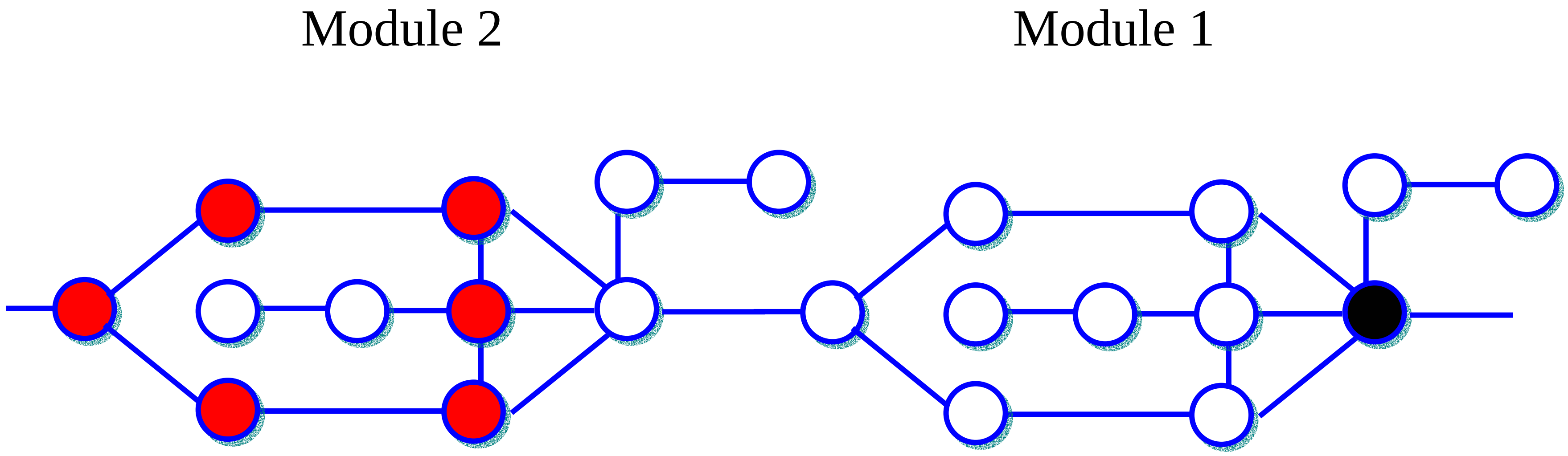
Module 2
Module 1

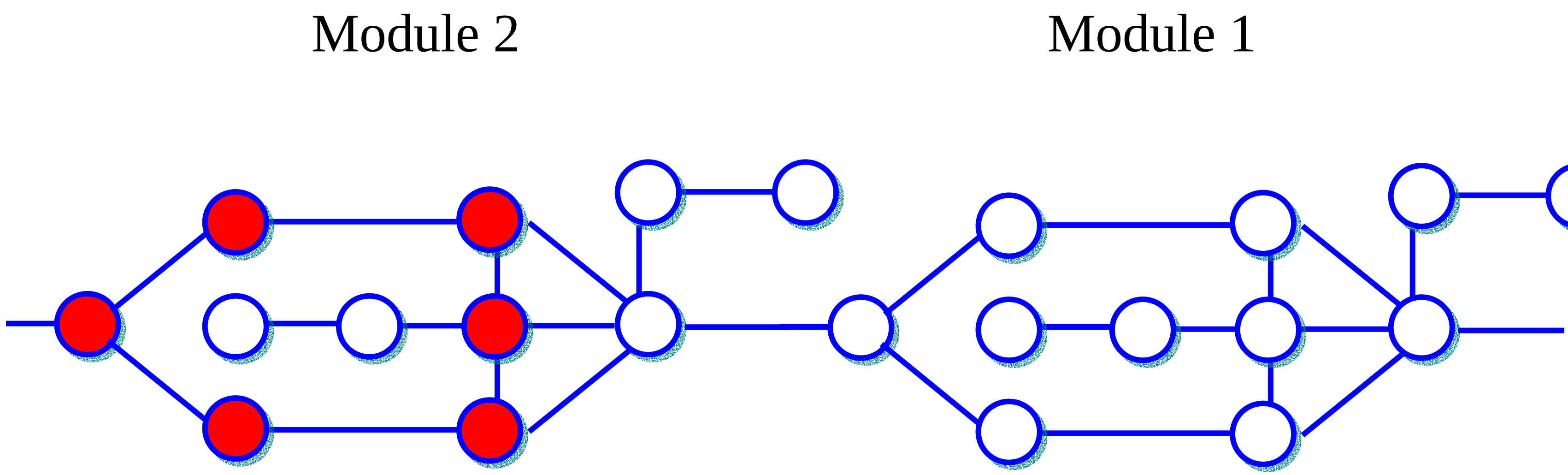

Module 2
Module 1

Module 2 Module 1




Module 2
Module 1


Module 2
Module 1



By induction on K we get desired.

Theorem 3.7. The local search problem (2–GGCP, *Flip*) is tightly PLS–reduced to the (p–median, *Swap*)