DM872

Mathematical Optimization at Work

Heuristic Solutions

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Outline

Heuristic Solutions

Different reasons may lead one to choose a heuristic:

- A solution is required rapidly, within a few seconds or minutes.
- The instance is so large and/or complicated that it cannot be formulated as an integer program (IP) or mixed integer program (MIP) of reasonable size.
- Even though it has been formulated as an MIP, it is difficult or impossible for the branch-and-cut system to find (good) feasible solutions.
- For certain combinatorial problems such as vehicle routing and machine scheduling, it is easy to find feasible solutions by inspection or knowledge of the problem structure, and a general-purpose MIP approach is ineffective.

Guarantees of Heuristics?

In designing and using a heuristic, there are various questions one can ask:

- Should one just accept any feasible solution, or should one ask a posteriori how far it is from optimal?
- Can one guarantee a priori that the heuristic will produce a solution within ϵ (or α%) of optimal?

(A couple of examples of heuristics with worst-case guarantees were given in DM871. Often the practical behavior of these heuristics is much better than the worst-case behavior, but is dominated in practice by some other heuristic.)

Primal Heuristics

Independent on MIP solvers

- Greedy heuristics
- Local search
- Metaheuristics

no dual bounds nor performance guarantee

Inside MIP solvers

- Construction heuristics
  - Rounding, Shift, Fix
  - Dive and Fix
  - Neighborhood Rounding
  - Feasibility pump
- Improvement heuristics
  - Local Branching
  - Proximity Search
  - Relaxation Induced Neighborhood Search (RINS)
  - Polishing Heuristic

User defined MIP heuristics

- Relax and Fix
- Large neighborhood search
- Extended formulations
- Cut and solve
- Problem specific heuristics

Outline

Greedy Heuristics

General combinatorial optimization problem:

$$\min_{S \subseteq N} \{c(S) \mid v(S) \geq k\}$$

example: knapsack
Assume: v is nondecreasing function, $v(N) \geq k$ and $c(\emptyset) = v(\emptyset) = 0$

A Greedy Heuristic

1. Set $S^0 = \emptyset$ (start with the empty set). Set $t = 1$.
2. Set $j_t = \arg\min \frac{c(S^{t-1} \cup \{j_t\}) - c(S^{t-1})}{v(S^{t-1} \cup \{j_t\}) - v(S^{t-1})}$ (choose the element whose additional cost per unit of resource is minimum).
3. If the previous solution S^{t-1} is feasible, i.e. $v(S^{t-1}) \geq k$, and $c(S^{t-1} \cup \{j_t\}) \geq c(S^{t-1})$, stop with $S^G = S^{t-1}$.
4. Otherwise set $S^t = S^{t-1} \cup \{j_t\}$.
5. If $t = n$, stop with $S^G = N$.
6. Otherwise set $t \leftarrow t + 1$, and return to 2.

Local search

Reformulation of combinatorial optimization problem as

$$\min_{S \subseteq N} \{c(S) \mid g(S) = 0\},$$

$g(S) \geq 0$ represents a measure of the infeasibility of set S.
Thus, the constraint $v(S) \geq k$ can be represented here by $g(S) = (k - v(S))^+$.

For a local search algorithm, we need to define:

- a solution,
- a local neighborhood structure $\mathcal{N}(S)$ for each solution $S \subseteq N$,
- an evaluation function $f(S)$, which can either be:
  1. just equal to $c(S)$ when S is feasible, and infinite otherwise, or
  2. a composite function of the form $c(S) + \alpha g(S)$ consisting of a weighted combination of the objective function value and a positive multiple α of the infeasibility measure for S.
- a search strategy (first improvement, simulated annealing, tabu search)

Outline

Construction Heuristics

$$\max\{cx \mid x \in X \subseteq \mathbb{Z}^n\}$$

generalizable to $\max\{cx \mid x \in X \subseteq \mathbb{Z}^n \times \mathbb{R}^p\}$.
Starting from an arbitrary point, the origin or some known point in the feasible LP region P:

1. Rounding. Select a variable x_j that should be integer and round to the nearest integer, denoted $\lceil x_j \rfloor$.
2. Shift. Shift x_j to $x_j \pm \alpha$ if this decreases infeasibility.
3. Fix. Set x_j to some value $x'_j \in \mathbb{Z}^1$ (using preprocessing or constraint programming to tighten bounds due to the fixing).

If repetition of these steps does not lead to a feasible solution,

- Solve a linear program with the variables that have been fixed and round the solution.

If this still does not lead to a feasible solution:

- After possibly fixing more variables based on the previous step, solve a mixed integer program for a limited amount of time.

Each of these steps is possibly repeated many times.

Dive-and-Fix

A Dive-and-Fix Heuristic

We suppose that we have a 0–1 problem. Given an LP solution x^*, let $F = \{j : x_j^* \notin \{0,1\}\}$ be the set of 0–1 variables that are fractional.

Initialization. Take the LP solution x^* at some node.

Basic Iteration. As long as $F \neq \emptyset$,

Let $i = \arg\min_{j \in F}\{\min[x_j^*, 1 - x_j^*]\}$ (find the variable closest to integer).

If $x_i^* < 0.5$, fix $x_i = 0$ (if close to 0, fix to 0).

Otherwise set $x_i = 1$ (if close to 1, fix to 1).

Solve the resulting LP.

If the LP is infeasible, stop (the heuristic has failed).

Otherwise let x^* be the new LP solution.

Termination. If $F = \emptyset$, x^* is a feasible mixed integer solution.

Neighborhood Rounding

The Neighborhood Rounding Heuristic

Given an optimal solution of the LP relaxation x^*, solve the IP

$$\max\{cx : x \in X, \lfloor x_j^* \rfloor \leq x_j \leq \lceil x_j^* \rceil \text{ for } j \in N\}.$$

Feasibility Pump

The Feasibility Pump Heuristic

Initialization. Let $x^* = x^{\text{LP}}$ be an optimal LP solution.

Iteration. Round to obtain an integer point $\hat{x} = \lceil x^* \rfloor$.

If $\hat{x} \notin P$, project onto the feasible region P.

If the IP is in the form $\max\{cx : Ax \leq b, \ell \leq x \leq k, x \in \mathbb{Z}^n\}$, the projection problem: $\min\{\sum_j |x_j - \hat{x}_j| : x \in P\}$ can be formulated as the linear program:

$$\begin{aligned} \min \sum_{j \in B : \hat{x}_j = \ell_j} (x_j - \ell_j) + \sum_{\in B : \hat{x}_j = k_j} (k_j - x_j) + \sum_{j : \ell_j < \hat{x}_j < k_j} \delta_j \\ Ax \leq b \\ \delta \geq x - \hat{x} \\ \delta \geq \hat{x} - x \\ \ell \leq x \leq k \end{aligned}$$

where variables $\delta_j = |x_j - \hat{x}_j|$ for all j are used to model the integer variables lying between their bounds. Let z be an optimal solution.

Set $x^* = z$ and repeat.

Improvement Heuristics

A Local Branching Heuristic

Given $x^* \in X$, solve an IP in which the feasible solutions lie in a small neighborhood of x^*. Specifically if $X \subseteq \{0,1\}^n$, solve the IP:

$$\max\left\{ cx : x \in X \cap \left\{ x : \sum_{j:x_j^*=0} x_j + \sum_{j:x_j^*=1} (1 - x_j) \leq k \right\} \right\},$$

where k is a small integer.

A Proximity Search Heuristic

Again we suppose that $x^* \in X \subset \{0,1\}^n$ is given. Selecting an improvement parameter $\delta > 0$, the idea is to find a feasible solution that improves on the best available solution by at least δ and is as close as possible to the starting solution x^*. This leads to the IP:

$$\min\Big(\sum_{j:x_j^*=0} x_j + \sum_{j:x_j^*=1} (1 - x_j)\Big)$$

$$\sum_{j=1}^{n} c_j x_j \leq cx^* - \delta$$

$$x \in X.$$

A Relaxation Induced Neighborhood Search (RINS) Heuristic

Given a feasible solution $x^* \in X$ and an optimal LP solution x^{LP}, let $F = \{j \in N : x_j^* = x_j^{\mathrm{LP}}\}$. Solve the IP:

$$\max\{cx : x \in X \cap \{x : x_j = x_j^* \text{ for } j \in F\}\}.$$

A Polishing Heuristic

Motivated by genetic algorithms, a list of at most k best solutions is stored. Two or more solutions $\{x^1, \dots, x^r\}$ are selected and then the idea is to fix the set $F = \{j \in [1, n] : x_j^1 = x_j^t \text{ for } t = 1, \dots, r\}$. Then again the IP

$$\max\{cx : x \in X \cap \{x : x_j = x_j^* \text{ for } j \in F\}\}$$

is solved. Here the selection of a pair of solutions is obtained by choosing a first solution randomly and then randomly choosing a second solution among the better solutions on the list. Mutations are introduced by randomly fixing only a subset of the variables in F.

Outline

User-Defined MIP heuristics

Relax-and-Fix Heuristic

Here we suppose that there is a partition of the variables and the problem can be written in the form:

$$Z = \max\{c^1x^1 + c^2x^2 : A^1x^1 + A^2x^2 = b, x^1 \in \mathbb{Z}_+^{n_1}, x^2 \in \mathbb{Z}_+^{n_2}\}.$$

The steps of the heuristic are then:

1. *Relax*. Solve the relaxation

$$\text{(MIP1)} \qquad \begin{aligned} \overline{Z} = \max\ & c^1x^1 + c^2x^2 \\ & A^1x^1 + A^2x^2 = b \\ & x^1 \in \mathbb{Z}_+^{n_1},\ x^2 \in \mathbb{R}_+^{n_2} \end{aligned}$$

   in which the integrality of the x^2 variables is dropped. Let $(\bar{x}^1, \bar{x}^2)$ be the corresponding solution.

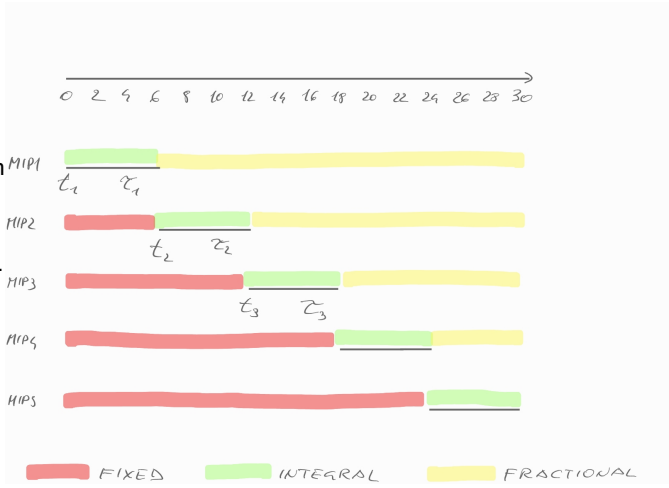
2. *Fix*. Fix the important variables x^1 at their values in MIP1, and solve the restriction

$$\text{(MIP2)} \qquad \begin{aligned} \underline{Z} = \max\ & c^1x^1 + c^2x^2 \\ & A^1x^1 + A^2x^2 = b \\ & x^1 = \bar{x}^1 \\ & x^2 \in \mathbb{Z}_+^{n_2}. \end{aligned}$$

   Let $(\tilde{x}^1, \tilde{x}^2)$ be the corresponding solution if MIP2 is feasible.
3. *Heuristic Solution*. The heuristic solution is $x^H = (\bar{x}^1, \tilde{x}^2)$ with $\underline{Z} = cx^H \leq Z \leq \overline{Z}$.

In practice, we often extend this idea by breaking up the set of variables into more than two subsets.

For instance, given a problem involving T time periods, we might break up the interval into K subintervals $[t_k, \tau_k]$ with $t_1 = 1, \tau_K = T$, and $t_{k+1} = \tau_k + 1$ for $k = 1, \ldots, K-1$ leading to a corresponding partition of the variables.



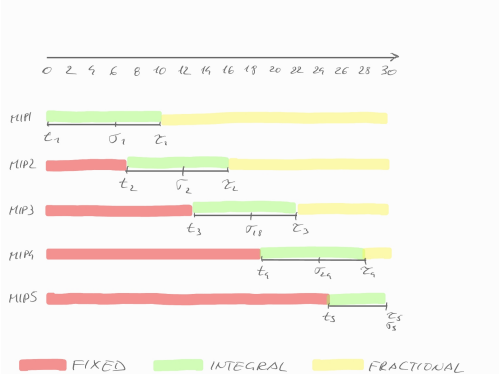
A more general version is obtained by working with triples $[t_k, \sigma_k, \tau_k]$ with $t_k \leq \sigma_k \leq \tau_k$, $t_k = \sigma_{k-1} + 1$. In the kth step, we solves the MIP

$$\max\{cx : x \in X, x_j = x_j^F \text{ for } j = 1, \dots, \sigma_{k-1},$$

$$x_j \in \mathbb{Z}_+^1 \text{ for } j = t_k, \dots, \tau_k, x_j \in \mathbb{R}_+^1 \text{ for } j > \tau_k\}$$

with optimal solution x^* and then set $x_j^F = x_j^*$ for $j = t_k, \dots, \sigma_k$.

Thus, for an instance with 30 periods in which we allow 10 variables to be integer in each restricted MIP and then fix the values of the first 6 integer variables, the corresponding values for the 5 MIPs are as follows: [1, 6, 10], [7, 12, 16], [13, 18, 22], [19, 24, 28], and [25, 30, 30].



Large Neighborhood Search Heuristic

This is a local search heuristic. To optimize over a large neighborhood either MIP or a specialized algorithm is used. Typically, the MIP is run until an improved solution is found or a time limit is reached. Given the problem in the form $\max\{cx : x \in X\}$ and a feasible solution $x^* \in X$, the basic idea is to define a subset of variables $V \subset N$ whose values it is hoped to improve by solving the MIP:

$$\max\{cx : x \in X, x_j = x_j^* \quad \text{for} \quad j \in N \setminus V\}.$$

For example after finding a feasible solution to the 30 period instance by Relax-and-Fix, one possibility is to set $V = [7, 12]$ to see if the solution between periods 7 and 12 can be improved. Alternatively, if the instance involved five items, we might fix the solution of items 1–4 and see if the solution of item 5 could

Extended Formulation Heuristics

Several problems can be formulated either in the original variables $\max\{cx : x \in P \cap \mathbb{Z}^n\}$ or with an extended formulation $\max\{cx + 0w : (x, w) \in Q \cap (\mathbb{Z}^n \times \mathbb{R}^p)\}$ where the first formulation provides very weak dual bounds, whereas the second provides good dual bounds, but is so large that it is only practical to solve its LP relaxation.

Here, the first option is to reduce the size of the extended formulation. Use the optimal LP solution $(x^{\text{LP}}, w^{\text{LP}})$ to fix sufficient variables $F_x \subseteq \{j \in [1, n] : x_j^{\text{LP}} \in \mathbb{Z}_+^1\}$ so that the restricted MIP

$$\max\{cx + 0w : x_j = x_j^{\text{LP}} \text{ for } j \in F_x, (x, w) \in Q \cap (\mathbb{Z}^n \times \mathbb{R}^p)\}$$

can be run to produce hopefully good feasible solutions.

A second option is to fix sufficient variables F_x so that the restricted original formulation

$$\max\{cx : x_j = x_j^{\text{LP}} \text{ for } j \in F_x, x \in P \cap \mathbb{Z}^n\}$$

produces good quality feasible solutions.

Cut and Solve

- Iteration $\equiv$ node in search path
- piercing cut a cut that removes at least one feasible solution from the original (unrelaxed) problem solution space.

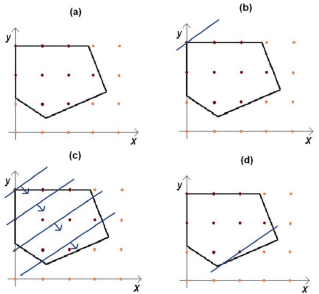
```
algorithm cut_and_solve (IP)
     select cut
     find optimal feasible solution in space removed by cut
        update best if necessary
     add cut to problem
     find lower bound
     if (lower bound >= best) return best
     otherwise, repeat
```

Example

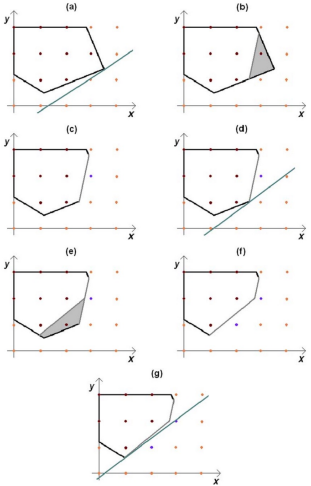
$$\min Z = y - \frac{4}{5}x$$

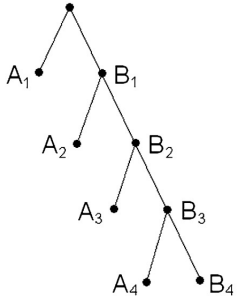
subject to:

$$x \geqslant 0$$
$$y \leqslant 3$$
$$y + \frac{3}{5}x \geqslant \frac{6}{5}$$
$$y + \frac{13}{6}x \leqslant 9$$
$$y - \frac{5}{13}x \geqslant \frac{1}{14}$$
$$x \in I$$
$$y \in I$$



Example





Generic piercing cut procedure

- Partition binary variables in a small set S and a large set L.
  - sparse problem solved on the set S while setting variables in L to zero.

$$\sum_{x_i \in L} x_i = 0$$

  - piercing cut

$$\sum_{x_i \in L} x_i \geq 1$$

- Assumption: being sparse in feasible (integer) solutions, this problem should be easier to solve.
- General guidelines to select S:
  - Each piercing cut should remove the solution to the current relaxed problem so as to prevent this solution from being found in subsequent iterations.
  - The space that is removed by the piercing cut should be adequately sparse, so that the optimal solution can be found relatively easily.
  - The piercing cuts should attempt to capture an optimal solution for the original problem. The algorithm will not terminate until an optimal solution has been cut away and consequently made the incumbent.
  - In order to guarantee termination, each piercing cut should contain at least one feasible solution for the original, unrelaxed, problem.

Generic Cut and Solve

```
algorithm generic_cut_and_solve (BIP)
     relax integrality and solve LP
     if (LP solution >= best) return best
     let S = {variables with reduced costs <= alpha}
     find optimal feasible solution in S
        update best if necessary
        if (LP solution >= best) return best
     add (sum of variables not in S >= 1) to BIP
     repeat
```

- Rationale: Reduced cost is a lower bound on the increase of the LP solution cost if the value of the variables is increased by one unit $\Longrightarrow$ variables with reduced costs of low absolute value are likely to disrupt the least the objective function value.
- α should be small enough to leave the sparse problem easy to solve and large enough to admit a feasible and possibly optimal solution
- note that since all variables currently in basis have reduced cost of null then the current optimal solution will be part of the sparse problem and cut away from the rest.

Asymmetric Traveling Salesman Problem

$$
\begin{aligned}
\text{(TSPIP)} \quad \min \quad & \sum_{ij \in A} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{ij \in \delta^+(i)} x_{ij} = 1 \qquad \text{for all } i \in V \\
& \sum_{ji \in \delta^-(i)} x_{ji} = 1 \qquad \text{for all } i \in V \\
& \sum_{ij \in A(S)} x_{ij} \leq |S| - 1 \qquad \text{for all } \emptyset \subset S \subset V, 2 \leq |S| \leq n-1 \\
& x_{ij} \in \{0,1\} \qquad \text{for all } ij \in E
\end{aligned}
$$

Relaxations:

- omit subtour elimination constraints $\rightsquigarrow$ Assignment Problem.
- relax integrality requirement with $0 \leq x_{ij} \leq 1, \forall i,j \in V$ (Held-Karp lower bound)

Cut and Solve for ATSP

- In the generic algorithm, lower bounds are found by relaxing integrality. For the ATSP, this is the Held-Karp lower bound.
- Then arcs with reduced costs less than α are selected for set S and a sparse graph composed of these arcs is solved. The best tour in this sparse graph becomes the first incumbent solution.
- Piercing cut: The original problem is then tightened by adding the constraint that the sum of the decision variables for the arcs in S is less than or equal to $n-1$, where n is the number of cities (at least one arc that is not in set S is required for an optimal tour).
- Repeat until the Held-Karp value of the deeply-cut problem is greater than or equal to the incumbent solution. At this point, the incumbent must be an optimal tour.
- After the first iteration, the incumbent solution can be used as an upper bound for the sparse problem solver, potentially improving its performance