Mathematical Programming as a Complement to Bio-Inspired Optimization

Ofer M. Shir (Tel-Hai College & Migal Institute, ISRAEL)

ofersh@telhai.ac.il

18th Int'l Conf. on Parallel Problem Solving from Nature
PPSN2024: September 2024, Hagenberg, AUSTRIA

about the presenter

Ofer Shir is an Associate Professor of Computer Science at Tel-Hai College and a Principal Investigator at the Migal Research Institute – Upper Galilee, ISRAEL.

Previously:

- IBM-Research
- Princeton University: Postdoctoral Research Associate
- PhD in CS: Leiden-U adv. : Th. Bäck & M. Vrakking; BSc in CS&Phys at Hebrew-U

why are we here?

- Global optimization has been for several decades addressed by algorithms and Mathematical Programming (MP) — branded as Operations Research (OR), yet rooted at Theoretical CS [1].
- Also – it has been treated by dedicated heuristics ("Soft Computing") – where EC resides (!)
- These two branches complement each other, yet practically studied under two independent CS disciplines

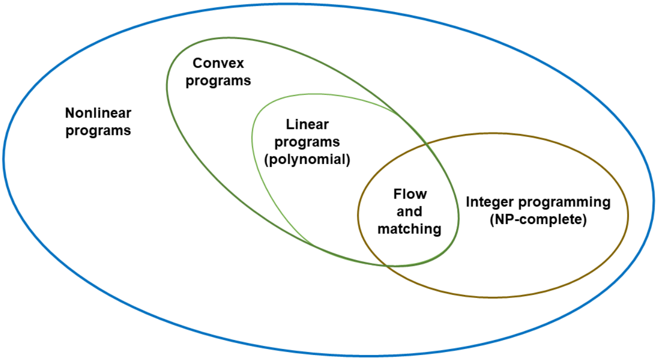
further motivation

EC scholars become stronger, better-equipped researchers when obtaining knowledge on this so-called "optimization complement"

Commonly-encountered **misbeliefs**:

- *"if the problem is non-linear, there is no choice but to employ a Randomized Search Heuristic"*
- *"if it's a combinatorial NP-complete problem, EAs are the most reasonable option to approach it"*
- *"neither Pareto optimization nor uncertainty is/are addressed by OR"*
- *"OR is the art of giving bad answers to problems, to which, otherwise worse answers are given"*

outline



Mathematical Programming: fundamentals

based on (i) MIT's "Optimization Methods" course material by D. Bertsimas,
(ii) "Combinatorial Optimization" by Ch. Papadimitriou & K. Steiglitz,
(iii) "The Nature of Computation" by C. Moore and S. Mertens, and
(iv) IBM's ILOG/OPL tutorials and documentation.

the field of operations research

- Developed during WW-II: mathematicians assisted the US-army to solve hard strategical and logistical problems; mainly planning of operations and deployment of military resources. Due to the strong link to military *operations*, the term *Operations Research* was coined.
- Post-war: knowledge transfer into industry
- Roots: linear programming (LP), pioneered by George B. Dantzig
- Dantzig worked for the US-government, formulating the generalized LP problem, and devising the Simplex algorithm for tackling it. He also pursued an academic career (Berkeley, Stanford).

mathematical optimization: CP $\cup$ MP

- Partitioning into 2 main approaches: constraints programming (CP) *versus* mathematical programming (MP). CP is concerned with constraints satisfaction problems (CSPs), which possess no objective functions (sometimes because impossible to model). CP is usually of little interest to us, but it is super important for **Formal Verification**, where tasks can be modeled as CSPs.
- MP includes the following techniques:
  1. linear programming (LP)
  2. integer programming (IP)
  3. mixed-integer programming (MIP)
  4. quadratic programming (QP) and mixed-integer QP (MIQP)
  5. nonlinear programming (NLP)

the canonical optimization problem

The general nonlinear problem formulated in the canonical form [2]:

$$\boxed{\begin{aligned} &\text{minimize}_{\vec{x}}\ f(\vec{x}) \qquad \vec{x} \in \mathbb{R}^d \\ &\text{subject to: } g_1(\vec{x}) \geq 0 \\ &\qquad\qquad\quad \vdots \\ &\qquad\qquad\quad g_m(\vec{x}) \geq 0 \\ &\qquad\qquad\quad h_1(\vec{x}) = 0 \\ &\qquad\qquad\quad \vdots \\ &\qquad\qquad\quad h_\ell(\vec{x}) = 0 \end{aligned}} \tag{1}$$

solving the general problem

- Convexity:
  1. $f : \mathcal{S} \rightarrow \mathbb{R}$
  2. The function is convex **iff** $\forall s_1, s_2 \in \mathcal{S}, \;\; 0 < \lambda < 1$

$$f\left(\lambda s_1 + (1-\lambda)\, s_2\right) \leq \lambda f\left(s_1\right) + (1-\lambda)\, f\left(s_2\right)$$

  3. f is concave if $-f$ is convex.
- The problem is called a *convex programming problem* when
  - i f is convex
  - ii g_i are all concave
  - iii h_j are all linear
- Strongest property: local optimality implies global optimality
- Sufficient conditions for optimality exist (Kuhn-Tucker)

linear programming: standard form

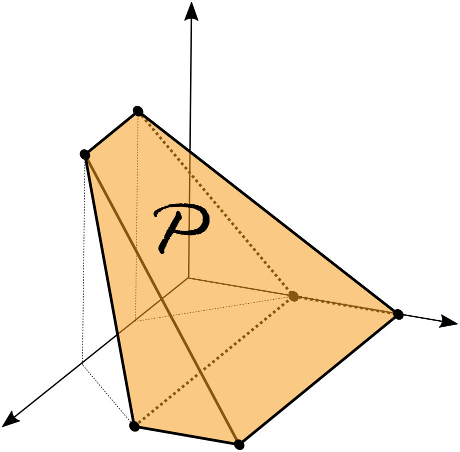
When f and the constraints are all linear, LP is formed by the **standard form** (minimization, equality constraints, non-negative variables) to search over a d-dimensional space, $\vec{x} \in \mathbb{R}^d$:

$$\boxed{\begin{aligned} &\text{minimize}_{\vec{x}}\ \vec{c}^T\vec{x} \\ &\text{subject to: } \mathbf{A}\vec{x} = \vec{b} \\ &\qquad\qquad\quad\ \vec{x} \geq 0 \end{aligned}} \tag{2}$$

with $\mathbf{A} \in \mathbb{R}^{m\times d}$ and $\vec{b} \in \mathbb{R}^m$ describing the constraints.

polyhedra

- A **hyperplane** is defined by the set
$$\left\{\vec{x} \in \mathbb{R}^d : \vec{a}^T\vec{x} = b_0\right\}$$
- A **halfspace** is defined by the set
$$\left\{\vec{x} \in \mathbb{R}^d : \vec{a}^T\vec{x} \geq b_0\right\}$$
- A **polyhedron** is constructed by the intersection of many halfspaces.
- The finite set of candidate solutions is the set of vertices of the **convex polyhedron** (*polytope*) defined by the linear constraints!
- Thus, solving any LP reduces to selecting a solution from a finite set of candidates $\Rightarrow$ the problem is **combinatorial** in nature.



geometry of LP

Given a *polytope*

$$\mathcal{P} := \left\{\vec{x} \in \mathbb{R}^d : \mathbf{A}\vec{x} \leq \vec{b}\right\}$$

- The point $\vec{x}$ is a **vertex** of $\mathcal{P}$
- $\vec{x} \in \mathcal{P}$ is an **extreme point** of $\mathcal{P}$ if

$$\nexists \vec{y}, \vec{z} \in \mathcal{P}\, (\vec{y} \neq \vec{x}, \vec{z} \neq \vec{x}) : \quad \vec{x} = \lambda \vec{y} + (1-\lambda)\, \vec{z},\ 0 < \lambda < 1$$

- $\vec{x} \geq \vec{0} \in \mathbb{R}^d$ is a **basic feasible solution** (**BFS**) iff $\mathbf{A}\vec{x} = \vec{b}$ and exist indices $\mathcal{B}_1, \ldots, \mathcal{B}_m$ such that:
  (i) the columns $\mathbf{A}_{\mathcal{B}_1}, \ldots, \mathbf{A}_{\mathcal{B}_m}$ are linearly independent
  (ii) if $\jmath \neq \mathcal{B}_1, \ldots, \mathcal{B}_m$ then $x_\jmath = 0$

polytopes and LP

"Corners" definitions: equivalence theorem

$\mathcal{P} := \left\{\vec{x} \in \mathbb{R}^d : \mathbf{A}\vec{x} \leq \vec{b}\right\}$; let $\vec{x} \in \mathcal{P}$.

$$\vec{x} \text{ is a vertex } \iff \vec{x} \text{ is an extreme point } \iff \vec{x} \text{ is a BFS}$$

See, e.g., [3] for the proof.

Conceptual LP search:

- begin at any "corner"
- **while "corner" is not optimal** hop to its neighbouring "corner" as long as it improves the objective function value

the basic simplex

$t \leftarrow 0$; *opt*, *unbounded* $\leftarrow$ `false`, `false`
$\vec{x}_t \leftarrow$ `constructBFS()`, $\mathbf{B} \leftarrow [\mathbf{A}_{\mathcal{B}_1}, \ldots, \mathbf{A}_{\mathcal{B}_m}]$
while *!opt && !unbounded* **do**
  if $\bar{c}_j := c_j - \vec{c}_B^T \mathbf{B}^{-1} \mathbf{A}_j \geq 0 \quad \forall j$ **then** *opt* $\leftarrow$ `true`
  else
    select any j such that $\bar{c}_j < 0$
    if $\vec{u} := \mathbf{B}^{-1} \mathbf{A}_j \leq \vec{0}$ **then** *unbounded* $\leftarrow$ `true`
    else
      $\vec{x}_{t+1} \leftarrow$ pivot on $\vec{x}_t$ `/* see [4] for details */`
      set new basis $\mathbf{A}_j$ `/* see [4] for details */`
      $t \leftarrow t + 1$
    end
  end
end

output: $\vec{x}_t$

duality

i. Every LP has an associated problem known as its **dual**; min turns into max, each constraint in the primal has an associated dual variable:

$$\begin{aligned} &\text{minimize}_{\vec{x}} \quad \vec{c}^T\vec{x} \quad \vec{x}\in\mathbb{R}^d \\ &\text{subject to: } \mathbf{A}\vec{x}=\vec{b} \\ &\qquad\qquad\quad\ \vec{x}\geq 0 \end{aligned}$$

$$\begin{aligned} &\text{maximize}_{\vec{p}} \quad \vec{p}^T\vec{b} \quad \vec{p}\in\mathbb{R}^m \\ &\text{subject to: } \vec{p}^T\mathbf{A}\leq\vec{c}^T \end{aligned}$$

$$\begin{aligned} &\text{minimize}_{\vec{x}} \quad \vec{c}^T\vec{x} \quad \vec{x}\in\mathbb{R}^d \\ &\text{subject to: } \mathbf{A}\vec{x}\geq\vec{b} \end{aligned}$$

$$\begin{aligned} &\text{maximize}_{\vec{p}} \quad \vec{p}^T\vec{b} \quad \vec{p}\in\mathbb{R}^m \\ &\text{subject to: } \vec{p}^T\mathbf{A}=\vec{c}^T \\ &\qquad\qquad\quad\ \vec{p}\geq 0 \end{aligned}$$

ii. The dual of the dual is the primal.

duality theorems [von Neumann, Tucker]

- **Weak duality theorem**
  If $\vec{x} \in \mathbb{R}^d$ is primal feasible and $\vec{p} \in \mathbb{R}^m$ is dual feasible then

$$\vec{p}^T \vec{b} \leq \vec{c}^T \vec{x}$$

- Corollary: If $\vec{x}$ is primal feasible, $\vec{p}$ is dual feasible, and $\vec{p}^T \vec{b} = \vec{c}^T \vec{x}$, then $\vec{x}$ is optimal in the primal and $\vec{p}$ is optimal in the dual.

- **Strong duality theorem**
  Given an LP, if it has an optimal solution – then so does its dual – having equal objective functions' values.

$\Rightarrow$ **The dual provides a bound that in the best case equals the optimal solution to the primal – and thus can help solve difficult primal problems.**

dual simplex

- Simplex is a primal algorithm: maintaining primal feasibility while working on dual feasibility
- Dual-simplex: maintaining dual feasibility while working on primal feasibility –
  Implicitly use the dual to obtain an optimal solution to the primal as early as possible, regardless of feasibility; then hop from one vertex to another, while gradually decreasing the infeasibility while maintaining optimality
- **Dual-simplex is the first practical choice for most LPs**.

R. Vanderbei, *Linear Programming: Foundations and Extensions*. Springer, 5^{th} ed., 2020, ISBN: 978-3-030-39414-1.

simplex: convergence

- Dantzig's simplex finds an optimal solution to any LP in a finite number of steps (avoiding cycles is easy, but excluded here).
- Over half-century of improvements, its robust forms are very effective in treating very large LPs.
- However, simplex is not a polynomial-time algorithm, even if it is fast in practice over the majority of cases.
- *Pathological* LP-cases exist (e.g., the Klee-Minty cube [5]) – where an **exponential number of steps** is needed for convergence.
- An **ellipsoid algorithm** [5], devised by Soviet mathematicians in the late 1970's, is guaranteed to solve every LP in a polynomial number of steps.

“high-level” ellipsoid [Shor-Nemirovsky-Yudin]

input : a bounded convex set $\mathcal{P} \in \mathbb{R}^d$

$t \leftarrow 0$

$\mathcal{E}_t \leftarrow$ ellipsoid containing $\mathcal{P}$

while *center* $\vec{\xi}_t$ *of* $\mathcal{E}_t$ *is not in* $\mathcal{P}$ **do**

  let $\vec{c}^T\vec{x} \leq \vec{c}^T\vec{\xi}_t$ be such that $\left\{\vec{x} : \ \vec{c}^T\vec{x} \leq \vec{c}^T\vec{\xi}_t\right\} \supseteq \mathcal{P}$

  update to the ellipsoid with minimal volume containing the intersected subspace:

$$\mathcal{E}_{t+1} \leftarrow \mathcal{E}_t \cap \left\{\vec{x} : \ \vec{c}^T\vec{x} \leq \vec{c}^T\vec{\xi}_t\right\}$$

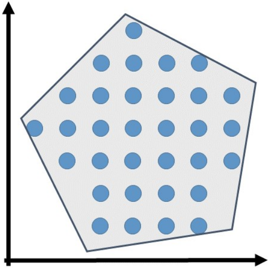
  $t \leftarrow t + 1$

end

output: *center* $\vec{\xi}_t \in \mathcal{P}$

ellipsoid aftermath

- Polynomial-time algorithm for obtaining $\vec{x}^*$ within any given bounded convex set
- Khachian first used it (1979) to show polynomial solvability of LPs
- **Theorem**: if there exists a polynomial-time algorithm for solving a strict linear inequalities problem, then there exists a polynomial-time algorithm for solving LPs (see [3] for the proof).
- Conceptual novelty: disregarding the combinatorial nature of LPs
- In practice, unlike simplex, the ellipsoid is slow yet steady.
- However, its theoretical "polynomiality" has strong implications also for discrete optimization.



discrete optimization

from LP to ILP

- The introduction of integer decision variables into a linear optimization problem yields a so-called (mixed)-integer linear program ((M)ILP) [6].
- A powerful modeling framework with much flexibility in describing discrete optimization problems
- The general ILP is itself *NP-complete* — and yet, there are subsets of “very easy” versus “very hard” problems
- *p2p shortest path* over a graph with d nodes has an algorithm with $\mathcal{O}(d^2)$ complexity, versus the *traveling salesman problem*...
- Unlike “pure-LP”, whose complexity is dictated by $d + m$ (variables+constraints), the choice of formulation in ILP is critical!
- Direction — what if the **constraint matrix** is **unimodular** [7] ?

integer linear optimization

- Pure integer:

$$\boxed{\begin{array}{rl} \text{maximize}_{\vec{x}} & \vec{c}^T\vec{x} \\ \text{subject to:} & \mathbf{A}\vec{x} \leq \vec{b} \\ & \vec{x} \in \mathbb{Z}_+^d \end{array}} \tag{3}$$

- Binary optimization (**important special case**):

$$(3) \text{ with } \vec{x} \in \{0,1\}^d$$

- Mixed-integer:

$$\boxed{\begin{array}{rl} \text{maximize}_{\vec{x}} & \vec{c}^T\vec{x} + \vec{h}^T\vec{y} \\ \text{subject to:} & \mathbf{A}\vec{x} + \mathbf{B}\vec{y} \leq \vec{b} \\ & \vec{x} \in \mathbb{Z}_+^d,\ \vec{y} \in \mathbb{R}_+^\ell \end{array}} \tag{4}$$

LP relaxations and the convex hull

- Given a discrete optimization problem, its consideration as a "*pure*" (continuous) LP is called its **LP relaxation**; e.g., each binary variable becomes continuous within the interval $[0, 1]$:

$$x_i \in \{0, 1\} \;\rightsquigarrow\; 0 \leq x_i \leq 1$$

- Formally, given a valid ILP formulation $\left\{\vec{x} \in \mathbb{Z}_+^d \mid \mathbf{A}\vec{x} \leq \vec{b}\right\}$, the polytope $\left\{\vec{x} \in \mathbb{R}^d \mid \mathbf{A}\vec{x} \leq \vec{b}\right\}$ constitutes its LP relaxation.
- The **convex hull** of a set of points is defined as the "smallest polytope" that contains all of the points in the set; given a finite set $S := \left\{p^{(1)}, \ldots, p^{(N)}\right\}$, it is defined as

$$\mathcal{C}(S) := \left\{ q \,\middle|\, q = \sum_{k}^{N} \lambda_k p^{(k)} \,,\; \sum_{k}^{N} \lambda_k = 1,\; \lambda_k \geq 0,\; p^{(k)} \in S \right\} \qquad (5)$$

- The **integral hull** is the *convex hull of the set of integer solutions*:

$$\widetilde{\mathcal{P}} := \mathcal{C}(X), \quad X \subset \mathbb{Z}^d \text{ solution points}$$

quality of formulations

- The quality of an ILP formulation for a problem having a feasible solution set X, is governed by the **closeness** of the *feasible set of its LP relaxation* to $\mathcal{C}(X)$.
- Given an ILP with two formulations, $\{P_1,\ P_2\}$, let $\left\{P_1^{LR},\ P_2^{LR}\right\}$ denote the feasible sets of their LP relaxations:
  we state that P_1 **is as strong as** P_2 if $P_1^{LR} \subseteq P_2^{LR}$, or that P_1 **is better than** P_2 if $P_1^{LR} \subset P_2^{LR}$ (strictly).
- If the *integral hull* is attainable as $\widetilde{\mathcal{P}} = \left\{\vec{x} \in \mathbb{R}^d \mid \widetilde{\mathbf{A}}\vec{x} \leq \widetilde{\vec{b}}\right\}$, the problem is polynomially solvable (all vertices are integers!) [6]
- *Another perspective*: an LP relaxation of an ILP with a **totally unimodular constraint matrix** has only integer solutions! [7]
- "**Easy Polyhedra**": MILP with fully-understood integral hulls — *assignment, min-cost flow, matching, spanning tree, etc.*

branch-and-bound

One of the common approaches to address integer programming, relying on the ability to bound a given problem.

It is a tree-search, adhering to the principle of *divide-and-conquer*:

(i) **branch**: select an active subproblem $\hat{\mathcal{F}}$

(ii) **prune**: if $\hat{\mathcal{F}}$ is infeasible – discard it

(iii) **bound**: otherwise, compute its lower bound $L(\hat{\mathcal{F}})$

(iv) **prune**: if $L(\hat{\mathcal{F}}) \geq U$, the current best upper bound, discard $\hat{\mathcal{F}}$

(v) **partition**: if $L(\hat{\mathcal{F}}) < U$, either completely solve $\hat{\mathcal{F}}$, or further break it to subproblems added to the list of active problems

“high-level” LP-based branch-and-bound

input : a linear integer program $\mathcal{F}$

$\Omega \leftarrow \{\mathcal{F}\}$; $U \leftarrow \infty$ `/* active problems' set; global upper bound */`

while Ω *is not empty* **do**

  let $\hat{\mathcal{F}}$ be a active subproblem, $\hat{\mathcal{F}} \in \Omega$; $\Omega \leftarrow \Omega \backslash \left\{\hat{\mathcal{F}}\right\}$

  compute its lower bound $L(\hat{\mathcal{F}})$ **by solving its LP relaxation**

  if $L(\hat{\mathcal{F}}) < U$ **then**

    $U \leftarrow L(\hat{\mathcal{F}})$

    if *exists heuristic solution* $\vec{\psi}$ *for* $\hat{\mathcal{F}}$ **then** $\vec{x}^* \leftarrow \vec{\psi}$

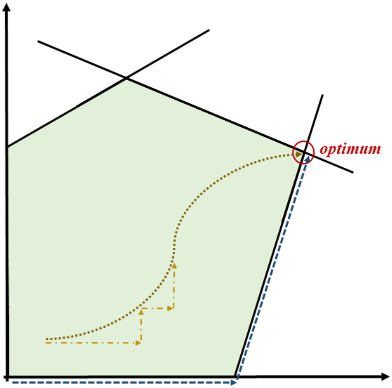
    else given the LP relaxation's optimizer, $\vec{\xi}$, if it contains a *fractional* decision variable ξ_i, construct 2 subproblems $\left\{\dot{\mathcal{F}}, \ddot{\mathcal{F}}\right\}$ by imposing either one of the new constraints $x_i \leq \lfloor \xi_i \rfloor$ or $x_i \geq \lceil \xi_i \rceil$ — and add them $\Omega \leftarrow \Omega \cup \left\{\dot{\mathcal{F}}, \ddot{\mathcal{F}}\right\}$

    `/* selection rules needed if #fractional` $\xi_i > 2$`*/`

  end

end

output: $\vec{x}^*$



MP in practice

obtaining an LP standard form

- LP's **standard form** (minimization, equality constraints, non-negative variables):

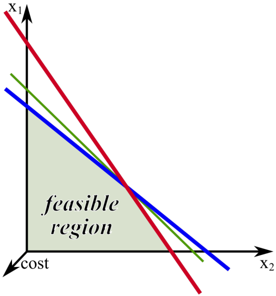
$$\boxed{\begin{array}{l} \text{minimize}_{\vec{x}}\ \vec{c}^T\vec{x} \\ \text{subject to: } \mathbf{A}\vec{x} = \vec{b} \\ \qquad\qquad\quad\ \vec{x} \geq 0 \end{array}}$$

- Applicable transformations to obtain standard form (introducing *slack/surplus* variables and accounting for *unrestricted* variables):

$$\begin{array}{lllll} \text{(a)} & \max\ \vec{c}^T\vec{x} & \Leftrightarrow & -\min\ \left(-\vec{c}^T\vec{x}\right) \\ \text{(b)} & \vec{a}_i^T\vec{x} \leq b_i & \Leftrightarrow & \vec{a}_i^T\vec{x} + s_i = b_i,\ s_i \geq 0 \\ \text{(c)} & \vec{a}_i^T\vec{x} \geq b_i & \Leftrightarrow & \vec{a}_i^T\vec{x} - s_i = b_i,\ s_i \geq 0 \\ \text{(d)} & -\infty < x_j < \infty & \Leftrightarrow & x_j := x_j^+ - x_j^-,\ \ x_j^+ \geq 0,\ x_j^- \geq 0 \end{array}$$

linear programming: solutions

$$\begin{aligned} &\text{minimize} \ \ -x_1 - x_2 \\ &\text{subject to: } x_1 + 2x_2 \leq 3 \\ &\qquad\qquad\qquad 2x_1 + x_2 \leq 3 \\ &\qquad\qquad\qquad x_1, x_2 \geq 0 \end{aligned}$$




```
dvar float+ x1,x2,s1,s2;
minimize
   -x1 - x2;
subject to {
    x1 + 2x2 + s1 == 3;
    2x1 + x2 + s2 == 3;
}
```

the fractional (continuous) knapsack problem

n items to be picked in a fractional way, $i = 1, \ldots, n$:

- v_i: value of each item
- w_i: weight of each item

Target: **maximize the total value** within a knapsack of capacity C.

$$\boxed{\begin{array}{l} \textbf{[FKP]} \quad \text{maximize} \sum_{i=1}^{n} v_i \cdot x_i \\ \text{subject to:} \\ \qquad \sum_{i=1}^{n} x_i \leq C \\ \qquad w_i \geq x_i \in \mathbb{R} \quad \forall i \in 1 \ldots n \end{array}} \tag{6}$$

basic f-knapsack in OPL

```
// Data reading from external database (or sheet or flat file)
{int} N = ...;
{float} CAPACITY = ...;
{float[N]} Values = ...;
{float[N]} Weights = ...;

dvar float+ select_ind[N] in 0..CAPACITY ;

maximize
   sum (n in N) (select_ind[n] * Values[n]) ;

subject to {
     forall (n in N) select_ind[n] <= Weights[n] ;
}
```

integer knapsack in OPL

```
// Data reading from external database
{int} N = ...;
{int} CAPACITY = ...;
{int[N]} Values = ...;
{int[N]} Weights = ...;

dvar int select_ind[N] in 0..1 ;

maximize
   sum (n in N) (select_ind[n] * Values[n]) ;

subject to {
      sum (n in N) select_ind[n]*Weights[n] <= CAPACITY;
}
```

solver operations

- Modern solvers allow the user to choose/tune their core algorithms:

```
cplex.startalg = 1; //primal simplex; for LP relaxation
cplex.lpmethod = 2; //dual simplex
cplex.epgap = 0.001; //relative MIP optimality gap
cplex.IntSolLim = 100; //number of integer solutions to stop
cplex.polishtime = 1800; //polishing time; see text below
cplex.tilim = 1800; //computation time limit
```

- Some MILP solvers actually employ *evolutionary operators* in their heuristic components, such as CPLEX's `polish` subroutine [8].

quadratic programming (QP)

- The simplest formulation of a QP has a *quadratic* objective function and *linear* constraints:

$$\boxed{\begin{array}{ll} \text{minimize}_{\vec{x}} & \frac{1}{2}\vec{x}^T\mathbf{Q}\vec{x} + \vec{c}^T\vec{x} \\ \text{subject to:} & \mathbf{A}\vec{x} \leq \vec{b} \\ & \vec{\ell} \leq \vec{x} \leq \vec{u} \end{array}} \tag{7}$$

- Renowned QP: the Markowitz portfolio – minimizing risk while ensuring minimal ROI, subject to a bounded portfolio investment:

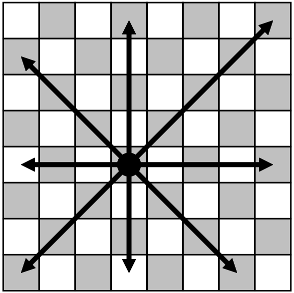
$$\begin{array}{l} \mathbf{Q}: \text{ portfolio's covariance matrix, representing RISK} \\ \vec{c} = \vec{0} \\ \vec{\rho}: \text{ stochastic return, representing ROI} \\ \text{constraints:} \quad \vec{\rho}^T\vec{x} \geq \text{ROI}_{min} \\ \qquad\qquad\quad \sum_i x_i = \text{INVEST}_{total} \end{array} \tag{8}$$

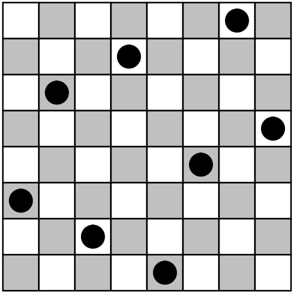
Markowitz: OPL implementation

```
{string} Investments = ...;
float Return[Investments] = ...;
float Covariance[Investments][Investments] = ...;
float BUDGET = ...;
float alpha = ...;

range float FloatRange = 0..BUDGET;
dvar float Allocation[Investments] in FloatRange;

maximize (sum(i in Investments) Return[i]*Allocation[i])
    - alpha*(sum(i,j in Investments)
        Covariance[i][j]*Allocation[i]*Allocation[j]);

subject to {
  // SPEND-IT-ALL: sum of allocations equals the given budget
  allocate: (sum (i in Investments) (Allocation[i])) == BUDGET;
}
```

CSP: the N-queens problem

The N-queens problem (NQP) [9] is defined as the task to place N queens on an $N \times N$ chessboard in such a way that they cannot *attack* each other.





N-queens as maximization

$$\text{maximize} \sum_{i,j} x_{ij}$$

subject to:

$$\sum_{i} x_{ij} \leq 1 \quad \forall j \in \{1 \ldots, N\}$$

$$\sum_{j} x_{ij} \leq 1 \quad \forall i \in \{1 \ldots, N\}$$

$$\sum_{j-i=k} x_{ij} \leq 1 \quad \forall k \in \{-N+2, -N+3, \ldots, N-3, N-2\}$$

$$\sum_{i+j=\ell} x_{ij} \leq 1 \quad \forall \ell \in \{3, 4, \ldots, 2N-3, 2N-1\}$$

$$x_{ij} \in \{0, 1\} \ \forall i, j \in \{1, \ldots, N\}$$

N-queens: OPL implementation [CSP]

```
int N = ...;
range R = 1..N;

dvar boolean queen [R][R];
// NO OBJECTIVE FUNCTION !
subject to {
  forall (s in R) {
      sum (t in R) queen[s][t] == 1;
     sum (t in R) queen[t][s] == 1;
   }
  forall (k in (-N+2)..(N-2)) {
      sum(s1 in R, t1 in R: t1-s1==k) queen[s1][t1] <= 1;
  }
  forall (k in 3..(2*N-1)) {
      sum(s1 in R, t1 in R: s1+t1==k) queen[s1][t1] <= 1;
  }
}
```

the traveling salesman problem

- The *archetypical* Traveling Salesman Problem (TSP) is posed as finding a Hamilton cycle of minimal total cost. Explicitly, given a directed graph G, with a vertex set $V = \{1, \dots, d\}$ and an edge set $E = \{\langle i, j \rangle\}$, each edge has cost information $c_{ij} \in \mathbb{R}^+$.

- **Black-box formulation: cyclic permutations**

$$\boxed{\begin{array}{l} \textbf{[TSP-perm]} \quad \text{minimize} \displaystyle\sum_{i=0}^{d-1} c_{\pi(i),\pi((i+1)_{\mathrm{mod}d})} \\ \text{subject to:} \\ \qquad \pi \in P_{\pi}^{(d)} \end{array}} \tag{9}$$

- But this is clearly not an MP, since it does not adhere to the canonical form!

ILP formulation [Miller-Tucker-Zemlin]

TSP as an ILP utilizes d^2 binary decision variables $\mathtt{x}_{ij}$:

$$\boxed{\begin{array}{l} \textbf{[TSP-ILP]} \ \ \text{minimize} \displaystyle\sum_{\langle i,j\rangle\in E} c_{ij}\cdot \mathtt{x}_{ij} \\ \text{subject to:} \\ \qquad \displaystyle\sum_{j\in V} \mathtt{x}_{ij} = 1 \quad \forall i\in V \\ \qquad \displaystyle\sum_{i\in V} \mathtt{x}_{ij} = 1 \quad \forall j\in V \\ \qquad \mathtt{x}_{ij}\in\{0,1\} \quad \forall i,j\in V \end{array}} \tag{10}$$

But is this enough? What about inner-circles?

ILP formulation [Miller-Tucker-Zemlin]

TSP as an ILP utilizes d^2 binary decision variables $\mathtt{x}_{ij}$:

$$\textbf{[TSP-ILP]} \quad \text{minimize} \sum_{\langle i,j\rangle \in E} c_{ij} \cdot \mathtt{x}_{ij}$$

subject to:

$$\begin{aligned} &\sum_{j\in V} \mathtt{x}_{ij} = 1 \quad \forall i \in V \\ &\sum_{i\in V} \mathtt{x}_{ij} = 1 \quad \forall j \in V \\ &\mathtt{x}_{ij} \in \{0,1\} \quad \forall i,j \in V \end{aligned} \tag{10}$$

But is this enough? What about inner-circles?

d integers $\mathtt{u}_i$ are needed as decision variables to prevent inner-circles:

$$\begin{aligned} &\ldots \\ &\mathtt{u}_i - \mathtt{u}_j + 1 \leq (d-1)\left(1-\mathtt{x}_{ij}\right) \quad \forall i,j \in 1\ldots d \\ &d \geq \mathtt{u}_i \geq 2 \quad \forall i \in \{2,3,\ldots,d\} \end{aligned} \tag{11}$$

the EC perspective

- Unlike GAs, which require dedicated mutation and crossover operators for cyclic permutations, the challenge here is mostly about obtaining an effective formulation
- Perhaps *counter-intuitively*, increasing the order of magnitude of constraints does not necessarily render the problem harder to be solved as MP.
- The given MTZ formulation for TSP is itself of a polynomial size; an alternative formulation possesses $\mathcal{O}\left(2^d\right)$ *subtour elimination constraints*, though **impractical for large graphs**.
- In any case, TSP's *integral hull* is unknown; an NP-hard problem.

- Note that EC researchers have started looking at TSP and other problems in a gray-box perspective: Darrell Whitley's tutorial on "Graybox Optimization and Next Generation Genetic Algorithms".

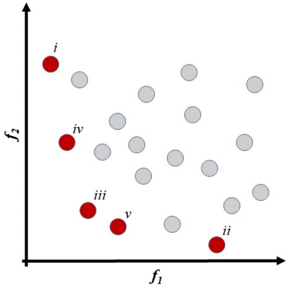
TSP on undirected graphs: OPL implementation

Addressing the undirected TSP by means of “node labeling” – assuming a single visit per node:

```
// Data preparation
tuple Raw_Edge {int point1; int point2; int dist; int active;}
{Raw_Edge} raw_edges = ...;

//Every edge is taken in both directions due to the graph
    nature, using ‘union’:
tuple Edge {int point1; int point2; int dist;}
{Edge} edges = {<e.point1, e.point2, e.dist> | e in raw_edges :
    e.active == 1}
      union {<e.point2, e.point1, e.dist> | e in raw_edges :
         e.active == 1};
{int} points = {e.point1 | e in edges};
int d = card (points); //set cardinality, i.e., number of cities
```

TSP in OPL continued: core model

```
dvar int edge_selector[edges] in 0..1;
dvar int label[points] in 0..d-1;

minimize sum (e in edges) edge_selector[e]*e.dist;

subject to {
  forall (p in points)
  ct_in_deg_equal_one:
    sum (e in edges : e.point2 == p) edge_selector[e] == 1;
  forall (p in points)
  ct_out_deg_equal_one:
    sum (e in edges : e.point1 == p)edge_selector[e] == 1;
  forall (e in edges : e.point2 != 1)
  ct_monotone_labeling:
    edge_selector [e] == 1 => label [e.point1] ==
        label[e.point2]-1;
}
```



extended topic: multiobjective optimization

multiobjective exact optimization

Diversity Maximization Approach (DMA) [10] key features:

- Iterative-exact nature: obtains a new **exact non-dominated solution** per each iteration
- Criteria exist for the attainment of the complete Pareto frontier
- Fine distribution of the existing set already found is guaranteed
- Optimality gap is provided – what may be gained by continuing constructing the Pareto frontier
- Solves any type of frontier (even if seems as a weighted sum)
- Importantly, DMA is **MILP if the original problem is MILP**

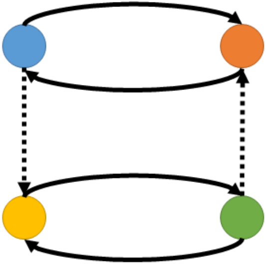
M. Masin and Y. Bukchin, 2008, “Diversity Maximization Approach for Multi-Objective Optimization”, *Operations Research*, 56, 411-424.

“high-level” DMA for M-objectives linear problems

input : a linear program featuring M objectives

1. Find an optimal solution for a weighted sum of multiple objectives with any reasonable strictly positive weights. If there is no feasible solution – **Stop**.
2. Set the partial efficient frontier equal to the found optimal solution. Choose optimality gap tolerance and maximal number of iterations.
3. If the maximal number of iterations is reached – **Stop**, otherwise **add M binary variables and $(M+1)$ linear constraints to the previous MILP model**.
4. Maximize the proposed diversity measure. If the diversity measure is less than the optimality gap tolerance – **Stop**, otherwise add the optimal solution to the partial efficient frontier and go to Step 3.

output: Pareto set, Pareto frontier



discussion

quick summary

- MP is a well-established domain encompassing a variety of algorithms with underlying rigorous theory.
- Broad knowledge of MP is valuable for both EC theoreticians and practitioners
- Given convex problems, MP is most likely the fittest tool
- Given discrete optimization problems that may be formulated as MILP/MIQP – it makes sense to first try MP-solvers
- MP is inherently adjusted to constrained problems (unlike EC...)
- Effective MP formulation lies in the heart of practical problem-solving
- Robustness to uncertainty, Pareto optimization, and hybridization are solid extensions to classical MP

O.M. Shir and M. Emmerich, 2024, “Multi-Objective Mixed-Integer Quadratic Models: A Study on Mathematical Programming and Evolutionary Computation”, *IEEE TREC*.

communities and resources

- INFORMS: The Institute for Operations Research and the Management Sciences; https://www.informs.org/
- COIN-OR: Computational Infrastructure for Operations Research – a project that aims to "create for mathematical software what the open literature is for mathematical theory"; https://www.coin-or.org/
- MATHEURISTICS: model-based metaheuristics, exploiting MP in a metaheuristic framework; http://mh2018.sciencesconf.org/

partial list of languages and solvers

- Modeling languages:
  1. GAMS
  2. AMPL
  3. OPL
  4. (`python` (Gurobi-Python, SciPy), MATLAB, ...)
- Environments and modeling systems:
  1. OR-Tools — Google Developers (open source!)
  2. IBM ILOG CPLEX (academia-free)
  3. Gurobi
  4. sas
  5. YALMIP
- Third-party solvers (free and open-source):
  1. CBC (via Coin-OR)
  2. GLPK (GNU Linear Programming Kit)
  3. SoPlex
  4. LP_SOLVE

benchmarking and competitions

- MIPLIB: the Mixed Integer Programming LIBrary

  http://miplib.zib.de/

- CSPLib: a problem library for constraints

  http://csplib.org/

- SAT-LIB: the Satisfiability Library - Benchmark Problems

  http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

- TSP-LIB: the Traveling Salesman Problem sample instances

  http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Acknowledgements go to
Dr. Yossi Shiloach (IBM-Research retired)
Dr. Michael Masin (former IBM-Research)

danke

references

[1] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*. New York, NY, USA: John Wiley and Sons, 2011.

[2] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York: Cambridge University Press, 2004.

[3] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Dover Books on Computer Science, Mineola, NY, USA: Dover Publications, 1998.

[4] R. Vanderbei, *Linear Programming: Foundations and Extensions*. International Series in Operations Research & Management Science, Cham, Switzerland: Springer, fifth ed., 2020.

[5] C. Moore and S. Mertens, *The Nature of Computation*. Oxford, UK: Oxford University Press, 2011.

[6] A. Schrijver, *Theory of Linear and Integer Programming*. Chichester, England: John Wiley and Sons, 1998.

[7] J. Matousek and B. Gärtner, *Understanding and Using Linear Programming*. Universitext, Springer Berlin Heidelberg, 2007.

[8] E. Rothberg, “An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions,” *INFORMS Journal on Computing*, vol. 19, no. 4, pp. 534–541, 2007.

[9] J. Bell and B. Stevens, “A survey of known results and research areas for n-queens,” *Discrete Math.*, vol. 309, pp. 1–31, Jan. 2009.

[10] M. Masin and Y. Bukchin, “Diversity maximization approach for multiobjective optimization,” *Operations Research*, vol. 56, no. 2, pp. 411–424, 2008.