Multi-objective Integer Programming: An Improved Recursive Algorithm

Melih Ozlen, School of Mathematical and Geospatial Sciences
RMIT University, Australia

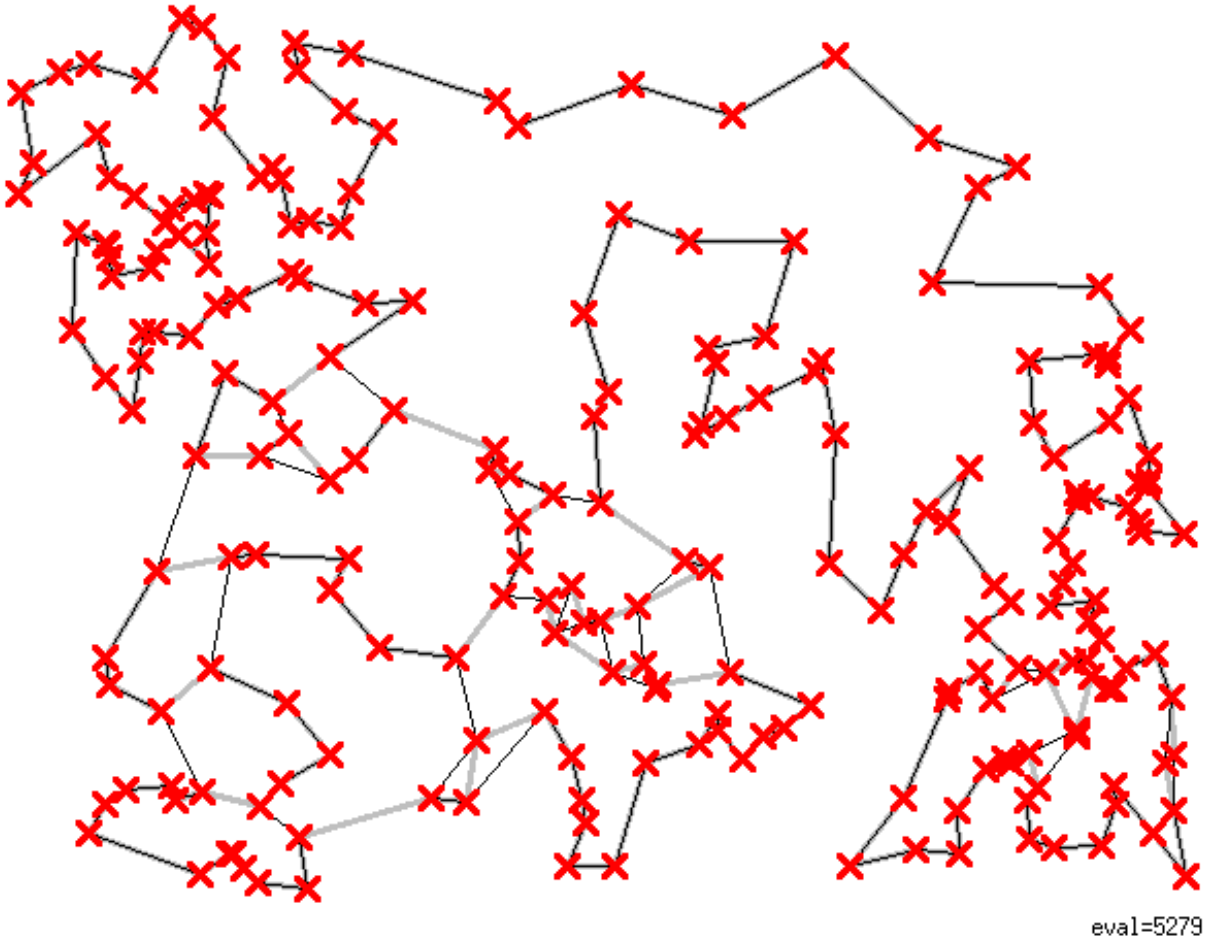
Benjamin A. Burton, School of Mathematics and Physics,
The University of Queensland, Australia

Outline

- Motivation
- Multi-Objective Optimization
- Bi-Objective Integer Programming
- Multi-Objective Integer Programming
- Results
- Conclusion
- Future Research

Motivation

Travelling Salesperson Problem (TSP)



Motivation

- Travelling Salesperson Problem
- *n* nodes to cover
- cost matrix from node *i* to node *j*
- *time* matrix from node *i* to node *j*
- find the minimum cost of covering all nodes
- minimum cost is $50k
- find the minimum duration of covering all nodes
- minimum duration is 100 minutes

Motivation

Objective	Constraint	Cost ($k)	Time (mins)
Minimise Cost		50*	100
Minimise Time		70	80*

Integer Programming

- Logistics/Production planning
- Scheduling
- Network design
- Workforce management
- Pure Maths
  - Computational Geometry
  - Topology

Multiple Objectives

- Cost/Profit
- Environment impact
- Risk/Safety
- Sustainability
- Waste

Multi-Objective Integer Programming

k objectives

$$\min\ f_1(x) = \sum_{j=1}^{n} c_{j1} x_j,\ f_2(x) = \sum_{j=1}^{n} c_{j2} x_j, \ldots,\ f_k(x) = \sum_{j=1}^{n} c_{jk} x_j$$

Integer objective coefficients

$$c_{j1}, c_{j2}, \ldots, c_{jk} \in \mathbb{Z}$$

Problem constraints

$$x \in X, x \geq 0, x \in \mathbb{Z}$$

Multi-Objective Optimization

For a multi-objective optimization problem,

aiming to minimize all objectives, defined as,

Min $f_1(x), f_2(x), \ldots f_k(x),$ st. $x \in X$,

a solution $x' \in X$ is **efficient** if and only if

there is no $x \in X$ such that $f_i(x) \leq f_i(x')$ for all $i \in \{1, \ldots, k\}$

and $f_i(x) < f_i(x')$ for at least one i. The resulting

objective vector $f(x')$ is said to be **nondominated.**

Multi-Objective Optimization

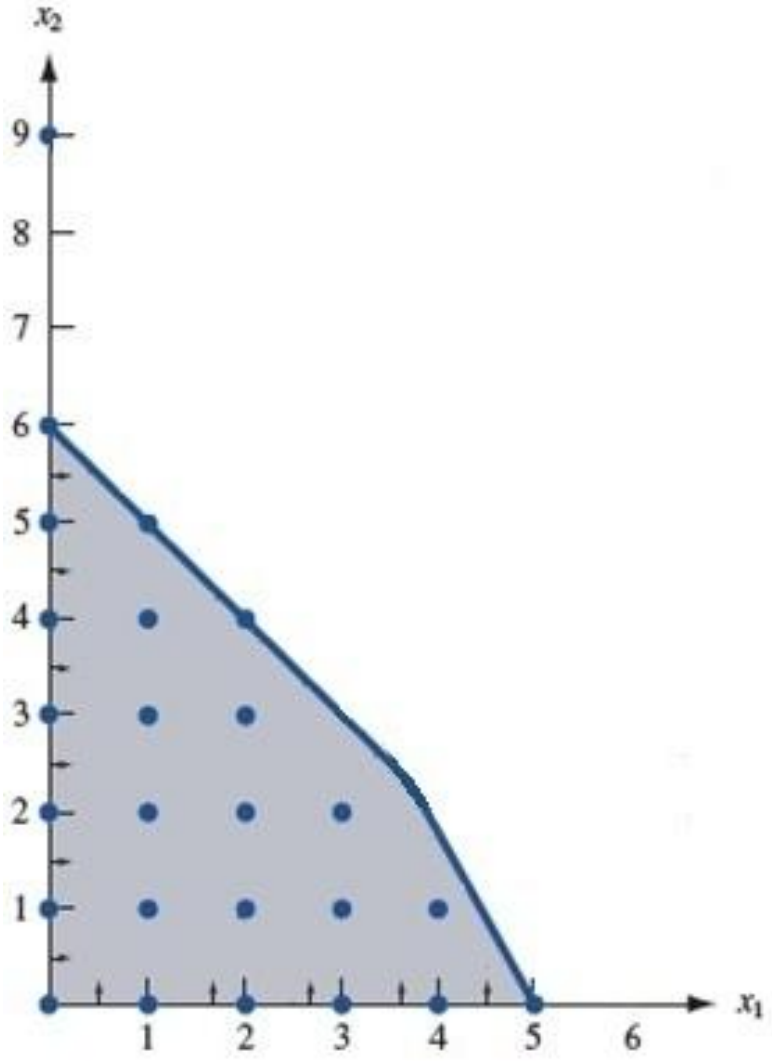
An efficient solution $x \in X$ is **supported efficient**,

if it minimizes some convex combination of

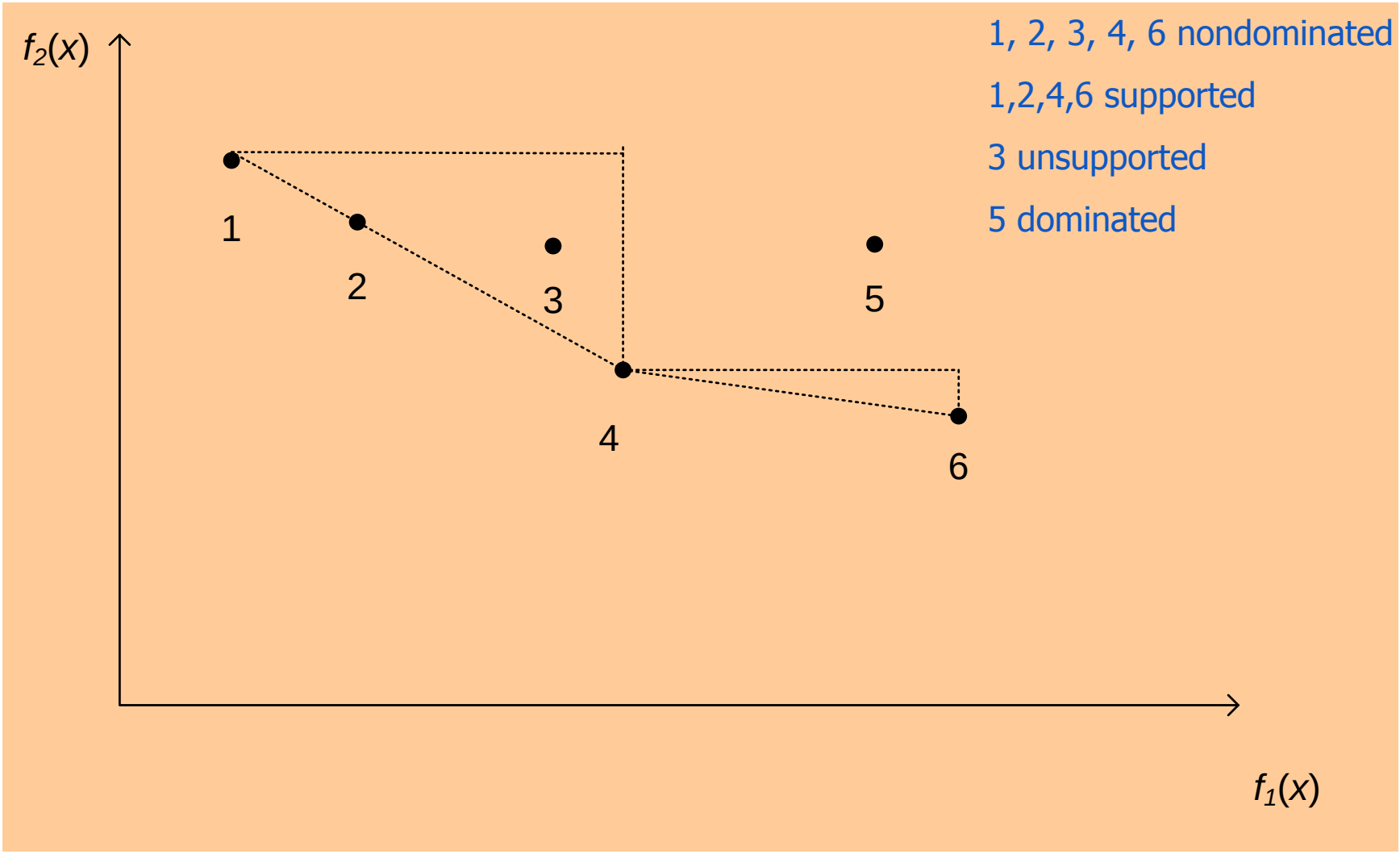
$f_1(x), f_2(x), \ldots f_k(x), \sum w_i f_i(x),$ where $w_i \geq 0, \sum w_i = 1.$

If there exists no such convex combination for an efficient solution, $x \in X$, then it is **unsupported efficient**.

Decision Space



Objective Space



Multi-Objective Integer Programming

- Hierarchical optimization
- Simultaneous optimization
  - Known utility function
    - Linear – find the best supported
    - Nonlinear – find the best nondominated
  - Unknown utility function
    - Linear – generate supported nondominated solutions
    - *Nonlinear – generate all nondominated solutions*

Multi-Objective Integer Programming

- Polynomially solvable
  - Assignment
  - Transportation/Transhipment
  - Network Flow
- NP-Hard
  - Bi-Objective Assignment
  - Bi-Objective Transportation
  - Bi-Objective Network Flow

Literature

- Known utility function
  - Linear
    - Abbas and Chaabane [2006], EJOR
    - Jorge [2009], EJOR
  - Nonlinear
    - Ozlen et al. [2010]
- Unknown utility function
  - Linear
    - Przybylski et al. [2010], IJOC
    - Özpeynirci and Köksalan [2010], MS

Literature

- Generating nondominated solutions
  - Klein and Hannah (1982), EJOR
    - Sequentially solving single objective problems
  - Sylva and Crema (2004), EJOR
    - Weighted objective to ensure generating all
  - Laumanns et al. (2006), EJOR
    - Adaptive ε-constraint method
  - Ozlen and Azizoglu (2009), EJOR
    - Recursive algorithm
  - Przybylski et al. (2010), DO
    - Two phase method
      - Supported
      - Unsupported

Bi-Objective Integer Programming

Two objectives

$$\min \; f_1(x) = \sum_{j=1}^{n} c_{j1} x_j, \; f_2(x) = \sum_{j=1}^{n} c_{j2} x_j$$

Integer objective coefficients

$$c_{j1}, c_{j2} \in \mathbb{Z}$$

Problem constraints

$$x \in X, x \geq 0, x \in \mathbb{Z}$$

Bi-Objective Integer Programming

Lexicographic optimization

$$\min\ f_1(x), \text{s.t. } x \in X$$

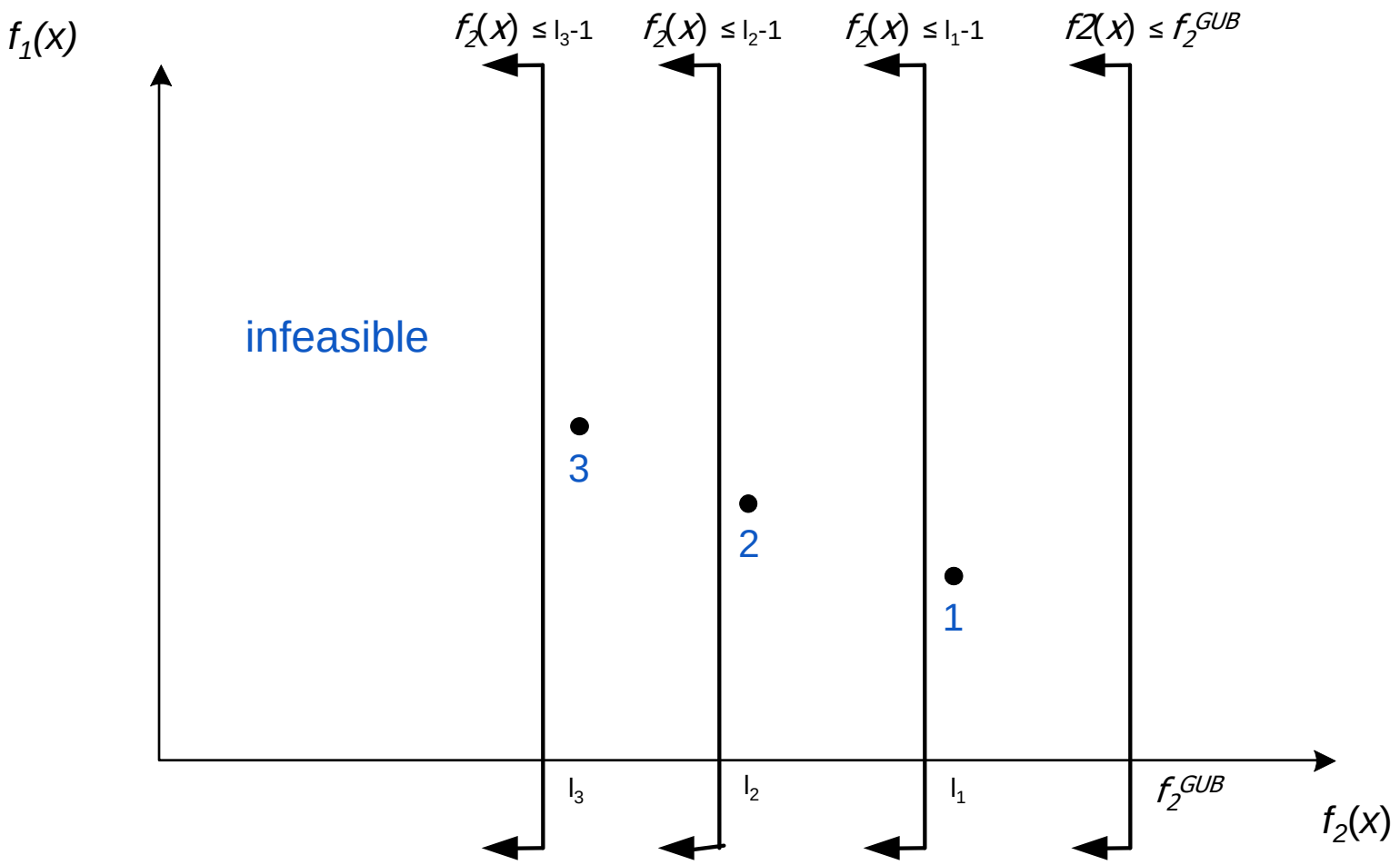
$$\min\ f_2(x), \text{s.t. } f_1(x) = f_1(x^*), x \in X$$

Constraint on the secondary objective

$$f_2(x) \le l$$

Initialize $l = \infty$, update as $l = f_2(x^*) - 1$, and stop when infeasible

Bi-Objective Integer Programming



Ozlen and Azizoglu (2009)

For any MOIP problem, a nondominated solution x' providing an upper bound on the k^{th} objective function value, $f_k(x)$, of all nondominated solutions, should also be nondominated with respect to all other $k-1$ objectives, $f_1(x), f_2(x), \ldots, f_{k-1}(x)$.

For any constrained MOIP problem, a nondominated solution x' providing an upper bound on the k^{th} objective function value, $f_k(x)$, of all nondominated solutions satisfying the constraints, should also be nondominated with respect to all other $k-1$ objectives, $f_1(x), f_2(x), \ldots, f_{k-1}(x)$.

Ozlen and Azizoglu (2009)

Constrained **L**exicographic **M**ulti **O**bjective **I**nteger **P**rogramming *

Lexicographic objective 1: Min $f_1(x), f_2(x),..., f_{k-1}(x)$

Lexicographic objective 1: Min $f_k(x)$

s.t.

$f_k(x) \leq l_k$

$x \in X$

f1(x)	f2(x)
10	20
14	18
19	12
25	9

Ozlen and Azizoglu (2009)

Algorithm 1 Özlen and Azizoğlu [2009]'s recursive algorithm for the original MOIP problem

Step 0. Set $l_k = \infty$ and initialise ND_k to the empty set.

Step 1. Solve the CLMOIP problem $(*)$, using Algorithm 1 to optimise the first $k-1$ objectives.
If the problem is infeasible then,
STOP.
Let the $(k-1)$-objective nondominated set be ND^*_{k-1}

Step 2. $ND_k = ND_k \cup ND^*_{k-1}$.
$l_k = \max\{f_k \mid f \in ND^*_{k-1}\} - 1$.
Go to Step 1.

Improved Recursive Algorithm

Intermediate problem

Lexicographic objective 1: Min $f_1(x), f_2(x),..., f_q(x)$

Lexicographic objective 1: Min $f_{q+1}(x)$

s.t.

$f_{q+1}(x) \le l_{q+1}, f_{q+2}(x) \le l_{q+2},...., f_k(x) \le l_k$

$x \in X$

We denote such a problem using the notation $(q, l_{q+1}, l_{q+2},..., l_k)$.

Improved Recursive Algorithm

- Let *P* be a CLMOIP problem, and let *R* be a relaxation of *P*. If *R* is infeasible, then *P* is also infeasible.
- Let *P* be a CLMOIP problem, and let *R* be a relaxation of *P*. If every nondominated objective vector for *R* is also feasible for *P*, then the set of all nondominated objective vectors for *P* is precisely the set of all nondominated objective vectors of *R*.
- If *R* has even a single nondominated objective vector that is not feasible for *P*, then the solution to *R* cannot be used to avoid solving *P*.

Improved Recursive Algorithm

The CLMOIP problem $(q, l'_{q+1}, l'_{q+2}, \ldots, l'_k)$ is a relaxation of $(q, l_{q+1}, l_{q+2}, \ldots, l_k)$ if $l'_i \geq l_i$ for all i and if $l'_i > l_i$ for some i.

Since Algorithm 1 iterates by incrementally lowering the bounds as the algorithm progresses it becomes highly likely that we can find a relaxation of the current CLMOIP amongst our set of already solved problems.

There could be many relaxations to a given CLMOIP, each with different nondominated sets, all of the relaxations should be examined until one is found that allows us to avoid solving the current CLMOIP.

Improved Recursive Algorithm

Algorithm 2 Improved recursive algorithm to generate nondominated set of MOIP

Step 0. Set $l_k = \infty$.

Step 1. Repeat:
Check the list of previously solved CLMOIPs to find a relaxation to the current CLMOIP problem $(*)$.
If all nondominated objective vectors for the relaxation are feasible for the current CLMOIP then,
Let that nondominated set be ND^*_{k-1} and go to Step 3.
If the relaxation is infeasible then,
STOP
Until there are no other relaxations to the current CLMOIP.

Step 2. Solve the CLMOIP problem $(*)$, using Algorithm 2 to optimise the first $k-1$ objectives.
If the problem is infeasible then,
STOP.
Let the $(k-1)$-objective nondominated set be ND^*_{k-1}

Step 3. $ND_k = ND_k \cup ND^*_{k-1}$.
$l_k = \max\{f_k \mid f \in ND^*_{k-1}\} - 1$.
Go to Step 1.

Improved Recursive Algorithm

- Coding complexities
  - Recursion or Stack Structure (B&B)
  - Data structures
    - Nondominated solutions
    - Solved problem information
  - Efficient Query methods
    - Locating useful relaxations

Example

3-obj	2-obj	1-obj		$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	Relaxation
$(3, \infty)$	$(2, \infty, \infty)$	$(1, \infty, \infty, \infty)$	*	11	19	12	14	
		$(1, 18, \infty, \infty)$	*	12	11	11	13	
		$(1, 10, \infty, \infty)$	*	13	9	16	11	
		$(1, 8, \infty, \infty)$	*	14	8	23	13	
		$(1, 7, \infty, \infty)$		inf				
	$(2, 22, \infty)$	$(1, \infty, 22, \infty)$		11	19	12	14	$(1, \infty, \infty, \infty)$
		$(1, 18, 22, \infty)$		12	11	11	13	$(1, 18, \infty, \infty)$
		$(1, 10, 22, \infty)$		13	9	16	11	$(1, 10, \infty, \infty)$
		$(1, 8, 22, \infty)$		inf				
	$(2, 15, \infty)$	$(1, \infty, 15, \infty)$		11	19	12	14	$(1, \infty, \infty, \infty)$
		$(1, 18, 15, \infty)$		12	11	11	13	$(1, 18, \infty, \infty)$
		$(1, 10, 15, \infty)$		inf				

Example (contd)

	$(2, 11, \infty)$	$(1, \infty, 11, \infty)$		12	11	11	13	
		$(1, 10, 11, \infty)$		inf				$(1, 10, 15, \infty)$
	$(2, 10, \infty)$	$(1, \infty, 10, \infty)$	*	15	16	7	12	
		$(1, 15, 10, \infty)$	*	16	15	10	13	
		$(1, 14, 10, \infty)$		inf				
	$(2, 9, \infty)$	$(1, \infty, 9, \infty)$		15	16	7	12	$(1, \infty, 10, \infty)$
		$(1, 15, 10, \infty)$		inf				
	$(2, 6, \infty)$	$(1, \infty, 6, \infty)$		inf				
$(3, 13)$	$(2, \infty, 13)$	$(1, \infty, \infty, 13)$		12	11	11	13	
		$(1, 10, \infty, 13)$		13	9	16	11	$(1, 10, \infty, \infty)$
		$(1, 8, \infty, 13)$		14	8	23	13	$(1, 8, \infty, \infty)$
		$(1, 7, \infty, 13)$		inf				$(1, 7, \infty, \infty)$

Experiment

- # objectives – 3 or 4
- Objective coefficient – U[1,10] or U[1,20]
- Problem size - 25, 100, 225 decisions
- 20 random instances under each setting
- # IPs solved
- Not solver dependent
- CPU time is not relevant

Results

cols × rows	c_{ij}	Tri-objective (TOIP)				
		$\|ND\|$	#IPs for Alg. 1	#IPs for Alg. 2	$\frac{\#\text{IPs Alg 2}}{\#\text{IPs Alg 1}}$	$\frac{\#\text{IPs Alg 2}}{\|ND\|}$
25 × 10	U[1, 10]	13	45	25	56%	1.9
	U[1, 20]	13	59	27	46%	2.1
100 × 20	U[1, 10]	115	409	176	43%	1.5
	U[1, 20]	158	899	268	30%	1.7
225 × 30	U[1, 10]	437	1343	609	45%	1.4
	U[1, 20]	681	3493	1070	31%	1.6

Results

cols × rows	c_i	Quad-Objective (QOIP)				
		$\lvert ND \rvert$	#IPs for Alg. 1	#IPs for Alg. 2	$\frac{\#\text{IPs Alg 2}}{\#\text{IPs Alg 1}}$	$\frac{\#\text{IPs Alg 2}}{\lvert ND \rvert}$
25 × 10	U[1, 10]	25	432	81	19%	3.2
	U[1, 20]	28	729	109	15%	3.9
100 × 20	U[1, 10]	732	14465	2044	14%	2.8
	U[1, 20]	1010	56679	3732	7%	3.7
225 × 30	U[1, 10]	5208	79408	12608	16%	2.4
	U[1, 20]	8086	385212	27080	7%	3.3

Conclusion

- Improved recursive algorithm solves significantly less IPs compared to the original.
- The average number of IPs solved in order to identify each nondominated solution is not sensitive to problem size.
- Average number of IPs solved increases with increasing number of objectives.

Current Research

- Parallel Implementation
  - Dividing over the objective value ranges
    - Works efficiently for large objective ranges
    - Almost perfect parallelisation
  - Starting with different objective orderings
    - Works well for tight objective ranges
    - Parallelisation efficiency drops with number of CPUs
- Open source and commercial solver integration

Future Research

- Lexicographic Integer Programming
  - Specialised efficient algorithms
    - B&B
    - B&C
- Multi-Objective Mixed Integer Programming
  - Hybrid with Multi-Objective Linear Programming
- Developing problem specific algorithms