MIT 2.680

UNMANNED MARINE VEHICLE AUTONOMY,
SENSING, AND COMMUNICATIONS

Lecture 13: Multi-Objective Optimization in the IvP Helm

Apr 21st 2026

Web: http://oceanai.mit.edu/2.680

Mike Benjamin, mikerb@mit.edu

Photo by Arjan Vermeij
GLINT '09

1

MITMECHE

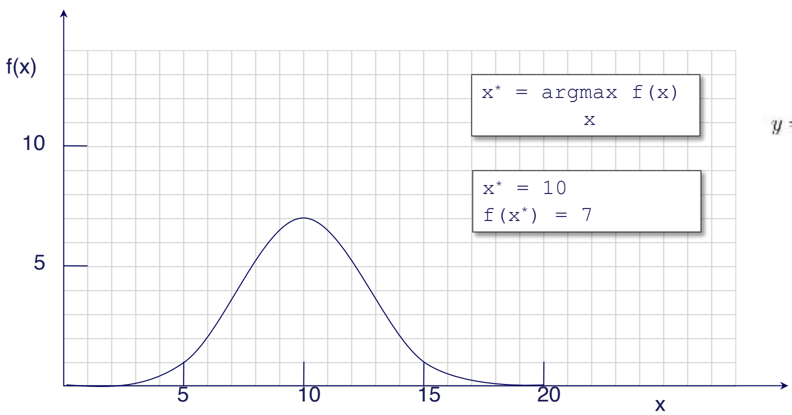
Outline

- What is an Optimization Problem?
- What is a Multi-Objective Optimization Problem?
- How is Multi-Objective Optimization Used on a Robot?
- Multi-Objective Optimization in the IvP Helm
- Introduction to the Tools for Creating Optimization Problems for the IvP Helm

2

MITMECHE

A Function with a Single Optima



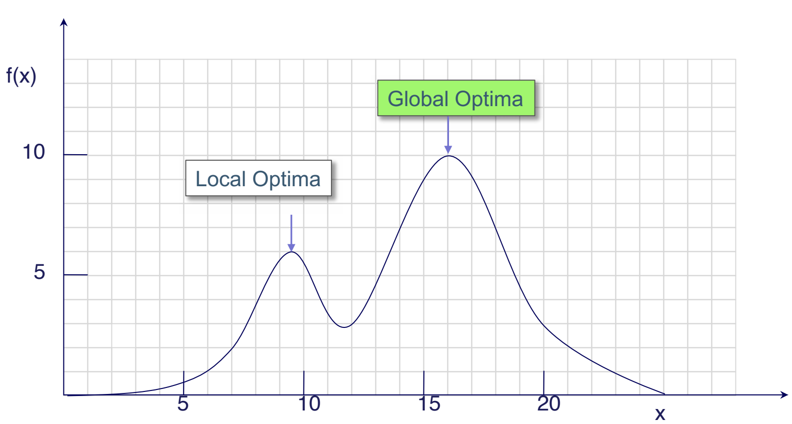
$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$\mu =$ Mean
$\sigma =$ Standard Deviation
$\pi \approx 3.14159\cdots$
$e \approx 2.71828\cdots$

3

MITMECHE

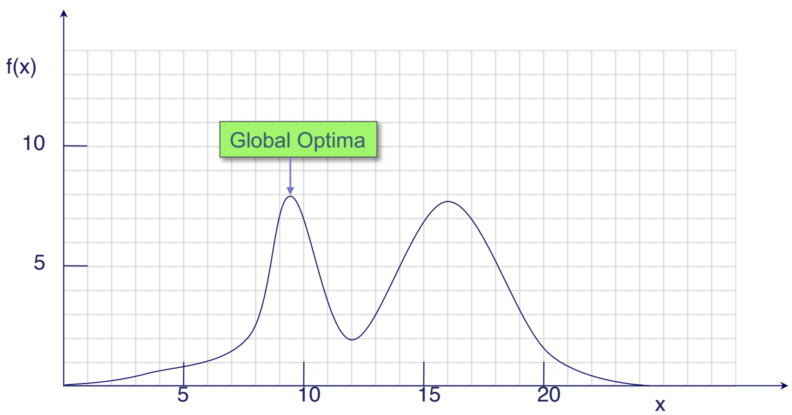
A Function with a Multiple Optima (“multi-modal” functions)



4

MITMECHE

Discrete Optimization



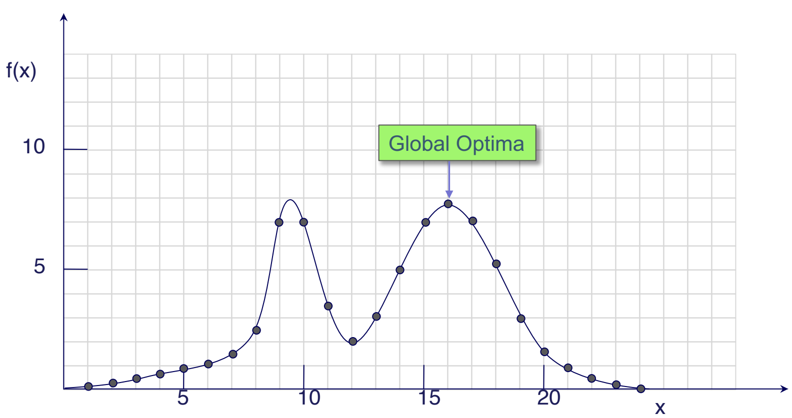
Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026 — MIT Dept of Mechanical Engineering

5

MITMECHE

Discrete Optimization



Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026 — MIT Dept of Mechanical Engineering

6

Mathematical Programming

Real World Problem

Problem Format (instance) → Solution Algorithm → Fast Solutions (with guaranteed properties)

7

Linear Programming

A simple example (from Ecker, Kupferschmid, 1988):

The Oakwood Furniture Company has 12.5 units of wood on hand from which to manufacture tables and chairs. Making a table uses two units of wood and making a chair uses one unit. Oakwood's distributor will pay $20 for each table and $15 for each chair, but he will not accept more than eight chairs and he wants at least twice as many chairs as tables. How many tables and chairs should the company produce to maximize revenue?

8

MITMECHE

Linear Programming

A simple example (from Ecker, Kupferschmid, 1988):

Objective Function: maximize $z = 20x_1 + 15x_2$ objective function

The Oakwood Furniture Company has 12.5 units of wood on hand from which to manufacture tables and chairs. Making a table uses two units of wood and making a chair uses one unit. Oakwood's distributor will pay $20 for each table and $15 for each chair, but he will not accept more than eight chairs and he wants at least twice as many chairs as tables. How many tables and chairs should the company produce to maximize revenue?

MITMECHE

Linear Programming

A simple example (from Ecker, Kupferschmid, 1988):

Objective Function: maximize $z = 20x_1 + 15x_2$ objective function

subject to:

$$\begin{aligned} & x_2 \le 8 \\ 2x_1 - & x_2 \le 0 \\ 2x_1 + & x_2 \le 12.5 \\ x_1 & \ge 0 \\ & x_2 \ge 0 \end{aligned}$$

constraints

The Oakwood Furniture Company has 12.5 units of wood on hand from which to manufacture tables and chairs. Making a table uses two units of wood and making a chair uses one unit. Oakwood's distributor will pay $20 for each table and $15 for each chair, but he will not accept more than eight chairs and he wants at least twice as many chairs as tables. How many tables and chairs should the company produce to maximize revenue?

MITMECHE

Linear Programming

A simple example (from Ecker, Kupferschmid, 1988):

Objective Function: maximize $z = 20x_1 + 15x_2$ — objective function

subject to:

$$
\begin{aligned}
x_2 &\le 8 \\
2x_1 - x_2 &\le 0 \\
2x_1 + x_2 &\le 12.5 \\
x_1 &\ge 0 \\
x_2 &\ge 0
\end{aligned}
\quad \text{constraints}
$$

The Oakwood Furniture Company has 12.5 units of wood on hand from which to manufacture tables and chairs. Making a table uses two units of wood and making a chair uses one unit. Oakwood's distributor will pay \$20 for each table and \$15 for each chair, but he will not accept more than eight chairs and he wants at least twice as many chairs as tables. How many tables and chairs should the company produce to maximize revenue?

MITMECHE

Linear Programming

A simple example (from Ecker, Kupferschmid, 1988):

Objective Function: maximize $z = 20x_1 + 15x_2$ — objective function

subject to:

$$
\begin{aligned}
x_2 &\le 8 \\
2x_1 - x_2 &\le 0 \\
2x_1 + x_2 &\le 12.5 \\
x_1 &\ge 0 \\
x_2 &\ge 0
\end{aligned}
\quad \text{constraints}
$$

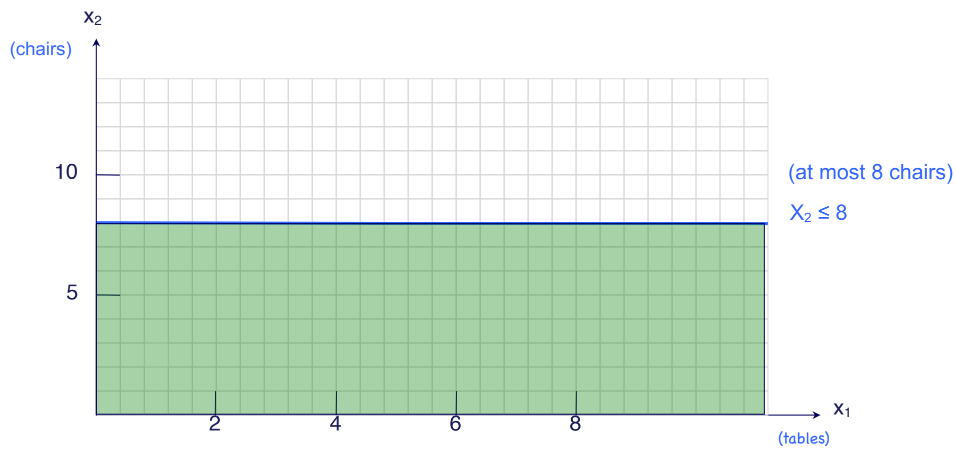
The Oakwood Furniture Company has 12.5 units of wood on hand from which to manufacture tables and chairs. Making a table uses two units of wood and making a chair uses one unit. Oakwood's distributor will pay \$20 for each table and \$15 for each chair, but he will not accept more than eight chairs and he wants at least twice as many chairs as tables. How many tables and chairs should the company produce to maximize revenue?

MITMECHE

Linear Programming

Feasible Space

The Oakwood Furniture Company Example (from Ecker, Kupferschmid, 1988)



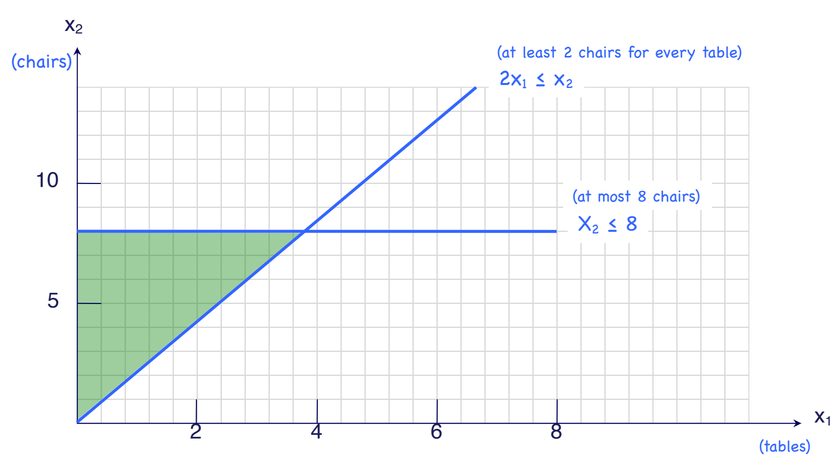
13

MITMECHE

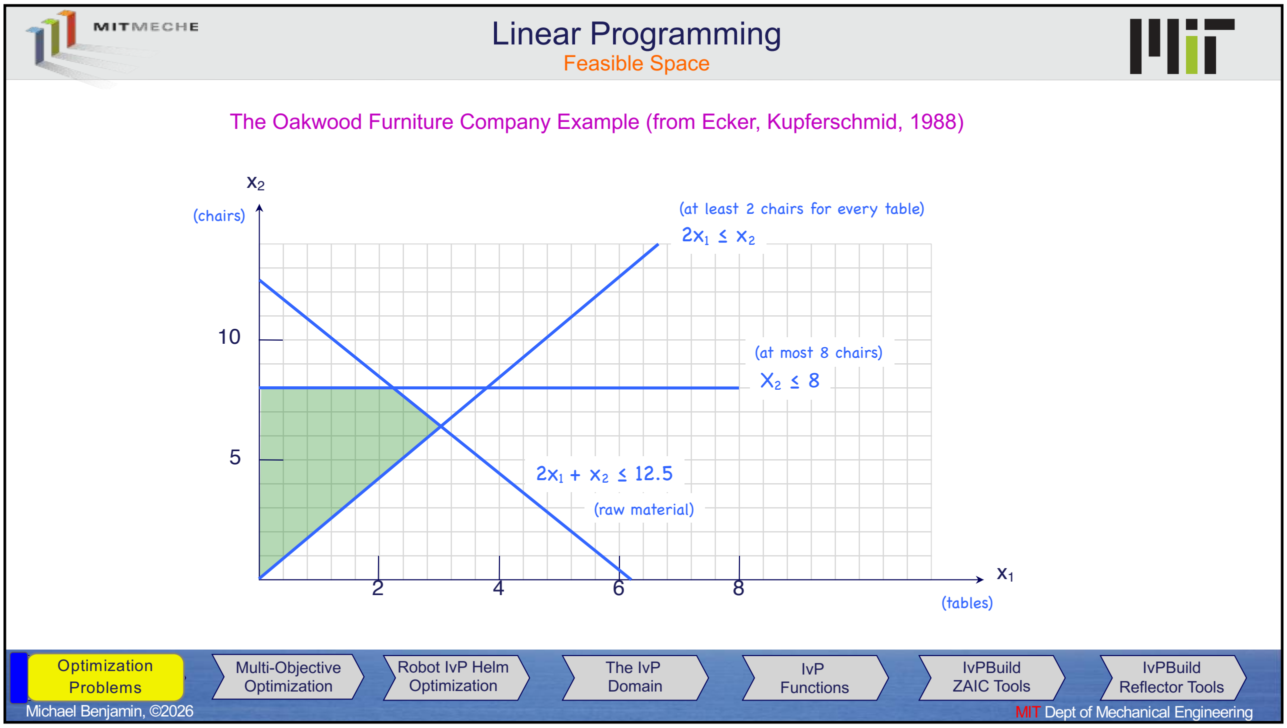
Linear Programming

Feasible Space

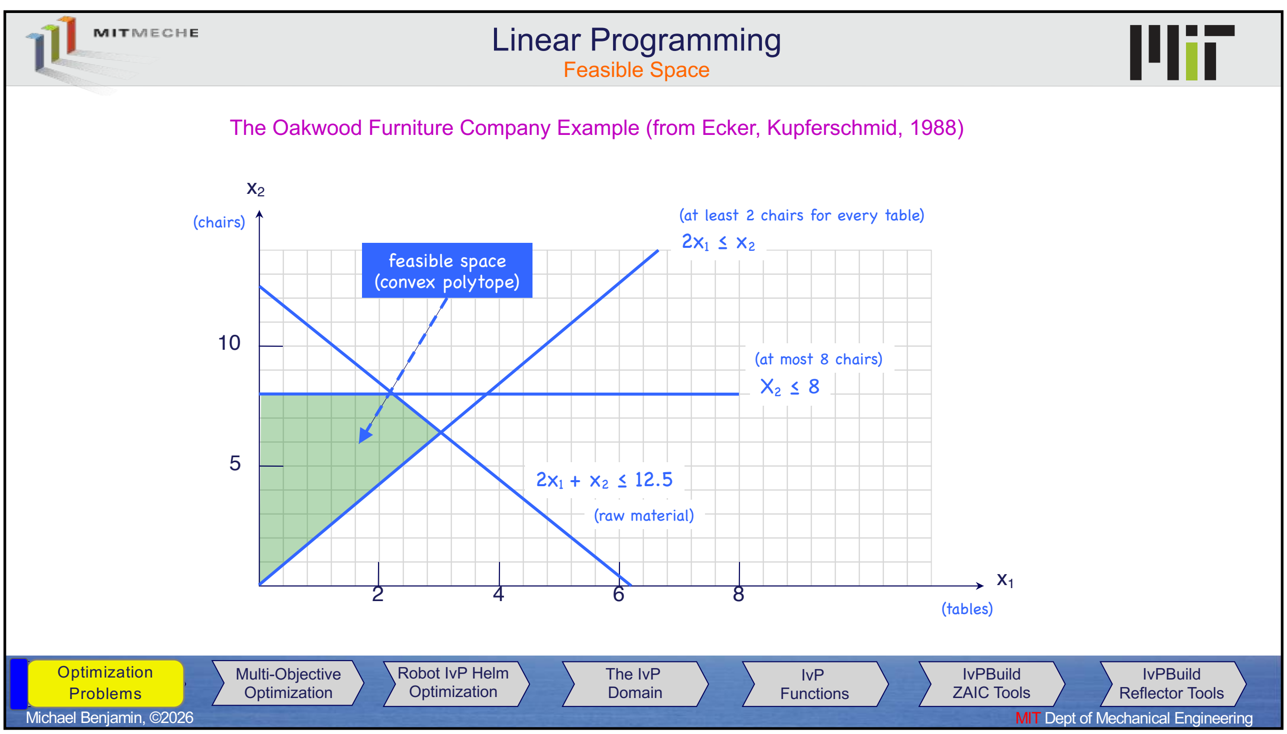
The Oakwood Furniture Company Example (from Ecker, Kupferschmid, 1988)



14


MITMECHE
Linear Programming
Feasible Space
The Oakwood Furniture Company Example (from Ecker, Kupferschmid, 1988)
x2
(chairs)
(at least 2 chairs for every table)
2x1 ≤ x2
10
(at most 8 chairs)
X2 ≤ 8
5
2x1 + x2 ≤ 12.5
(raw material)
2
4
6
8
x1
(tables)
Optimization Problems
Multi-Objective Optimization
Robot IvP Helm Optimization
The IvP Domain
IvP Functions
IvPBuild ZAIC Tools
IvPBuild Reflector Tools
Michael Benjamin, ©2026
MIT Dept of Mechanical Engineering

15


MITMECHE
Linear Programming
Feasible Space
The Oakwood Furniture Company Example (from Ecker, Kupferschmid, 1988)
x2
(chairs)
feasible space
(convex polytope)
(at least 2 chairs for every table)
2x1 ≤ x2
10
(at most 8 chairs)
X2 ≤ 8
5
2x1 + x2 ≤ 12.5
(raw material)
2
4
6
8
x1
(tables)
Optimization Problems
Multi-Objective Optimization
Robot IvP Helm Optimization
The IvP Domain
IvP Functions
IvPBuild ZAIC Tools
IvPBuild Reflector Tools
Michael Benjamin, ©2026
MIT Dept of Mechanical Engineering

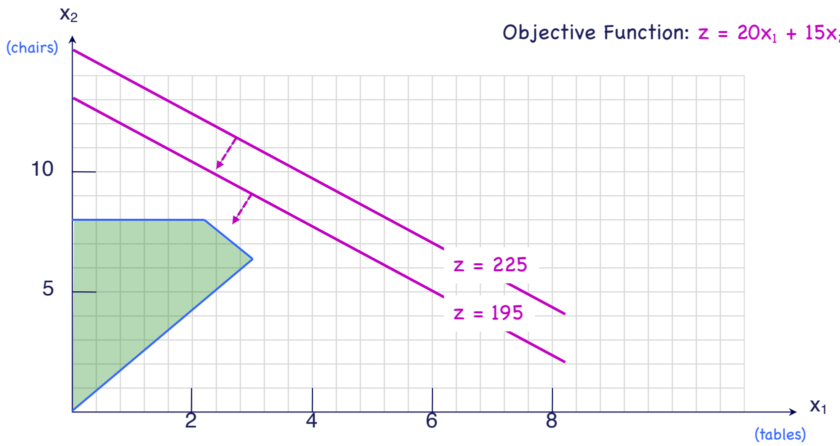
16

MITMECHE

Linear Programming

Graphical Solution Method

Graphical solution method: Only for illustration and works only in 2D.
If you could draw a line representing the objective function and "slide" it down the page until it hits a vertex...



Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026 — MIT Dept of Mechanical Engineering

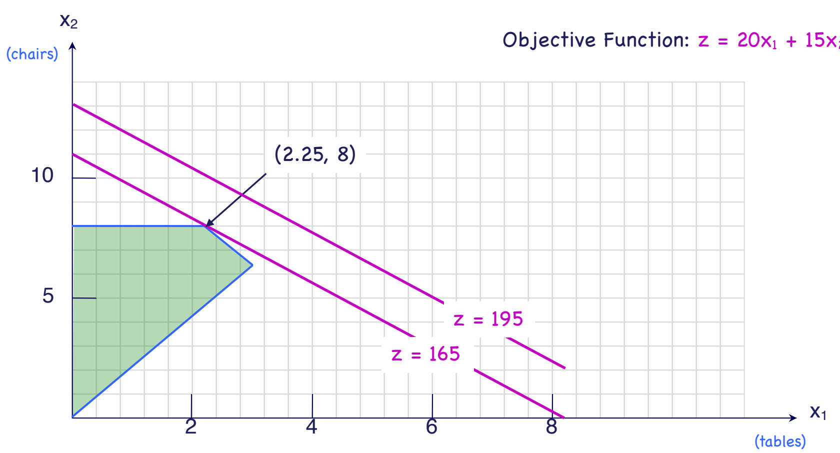
17

MITMECHE

Linear Programming

Graphical Solution Method

Graphical solution method: Only for illustration and works only in 2D.
If you could draw a line representing the objective function and "slide" it down the page until it hits a vertex...



Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026 — MIT Dept of Mechanical Engineering

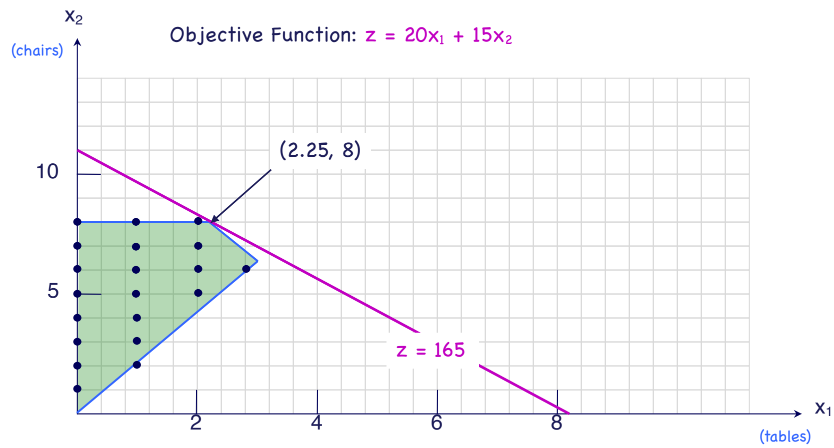
18

MITMECHE

Linear Programming

Integer vs. Continuous Solutions

- Note the optimal solution of 2.25 tables and 8 chairs.
- When the decision space is discrete, the problem may be better formulated as an integer programming problem.
- These problems are harder to solve generally – why?

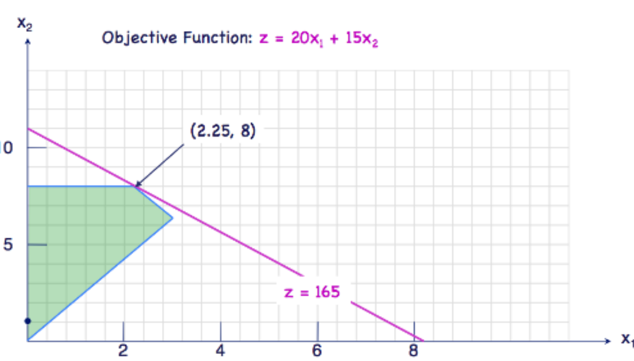


MITMECHE

Simplex Algorithm

Overview

- Capitalizes on the fact that the optimal solution resides at one of the vertices of the feasible space.
- The number of vertices in real-world problems is unmanageably large. It would not be practical to exhaustively investigate all vertices.
- Simplex proceeds from one vertex to another neighboring vertex, *always improving on the solution*. In practice it is very fast.

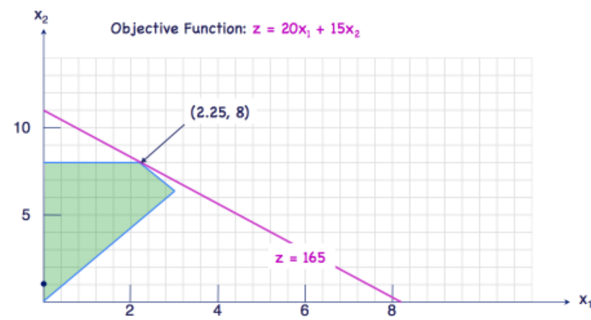


MITMECHE

Simplex Algorithm

Overview

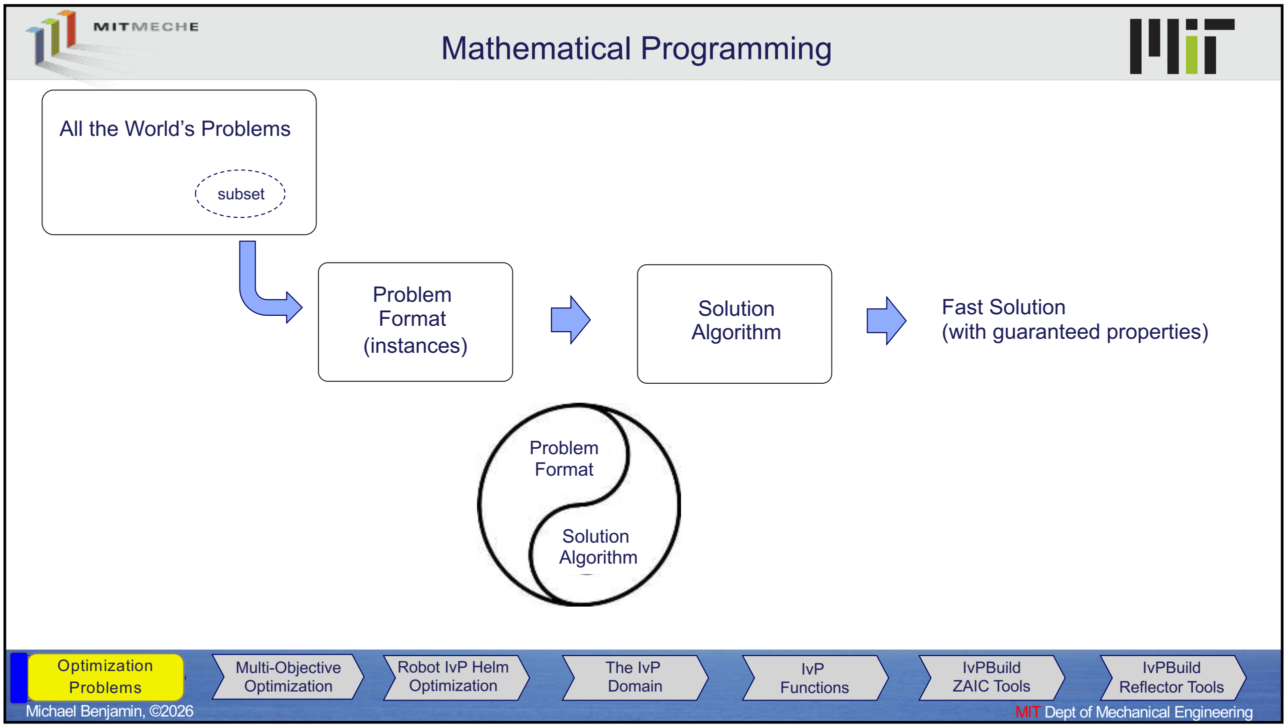
- Capitalizes on the fact that the optimal solution resides at one of the vertices of the feasible space.
- The number of vertices in real-world problems is unmanageably large. It would not be practical to exhaustively investigate all vertices.
- Simplex proceeds from one vertex to another neighboring vertex, *always improving on the solution*. In practice it is very fast.
- Simplex first published in 1948.
- The journal Computing in Science and Engineering listed it as one of the *top 10 algorithms of the twentieth century*.
- That being said, Simplex has been improved upon, including a class of algorithms referred to as *interior methods* that migrate from vertex to vertex more efficiently.



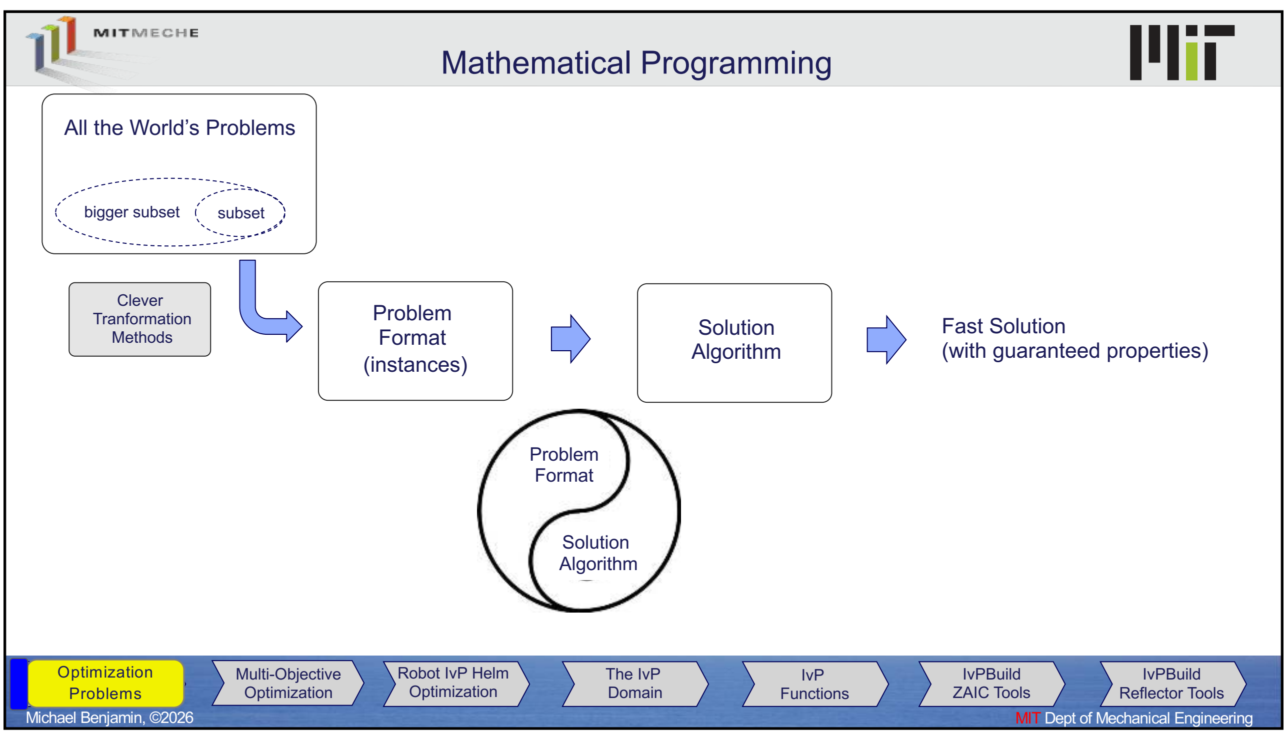
MITMECHE

The Key Idea of Mathematical Programming

1) The LP format has expressive power (Many real problems are LP problems)
2) The LP format may be exploited algorithmically (guaranteed fast solutions)


MITMECHE
Mathematical Programming
All the World's Problems
subset
Problem
Format
(instances)
Solution
Algorithm
Fast Solution
(with guaranteed properties)
Problem
Format
Solution
Algorithm
Optimization
Problems
Multi-Objective
Optimization
Robot IvP Helm
Optimization
The IvP
Domain
IvP
Functions
IvPBuild
ZAIC Tools
IvPBuild
Reflector Tools
Michael Benjamin, ©2026
MIT Dept of Mechanical Engineering

23


MITMECHE
Mathematical Programming
All the World's Problems
bigger subset
subset
Clever
Tranformation
Methods
Problem
Format
(instances)
Solution
Algorithm
Fast Solution
(with guaranteed properties)
Problem
Format
Solution
Algorithm
Optimization
Problems
Multi-Objective
Optimization
Robot IvP Helm
Optimization
The IvP
Domain
IvP
Functions
IvPBuild
ZAIC Tools
IvPBuild
Reflector Tools
Michael Benjamin, ©2026
MIT Dept of Mechanical Engineering

24

What it the relevance of Linear Programming
to Marine Autonomy?

MITMECHE

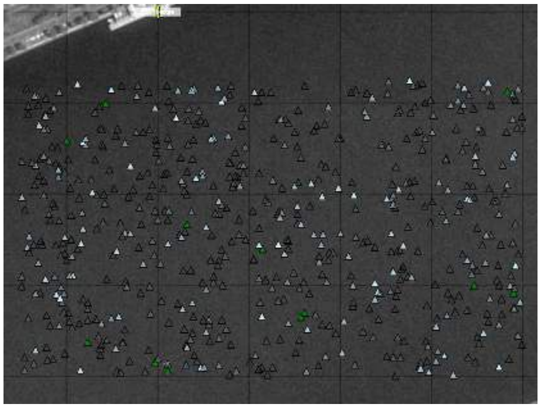
What it the relevance of Linear Programming
to Marine Autonomy?

Probably very little!

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools
Michael Benjamin, ©2026
MIT Dept of Mechanical Engineering

Optimization in Marine Autonomy

Consider the problem of Mine Search: Two vehicles search for hazardous objects, while minimizing the reporting of false alarms.



Optimization is may be *defined* by the stated problem parameters:

Objective Function: min $c_1x_1 + c_2x_2$
Constraint: s.t. $t < 9000$

where

- t is the total mission time in seconds
- x_1 is the number of missed hazards
- x_2 is the number of false alarms
- c_1 is 100 (the penalty for missing a hazard)
- c_2 is 35 (the false alarm penalty, for claiming a benign object as a hazard)

27

Optimization in Marine Autonomy

In **marine robotic platforms**, and real-world robots generally, decision making happens at several levels:

- Where and when is the next destination?
- What is our path plan?
- What are the sequence of heading and speed commands?
- What are the sequence of rudder and thrust commands?

Team Undergrads, 2018, MIT2.680 Hazard Search Lab

28

MITMECHE

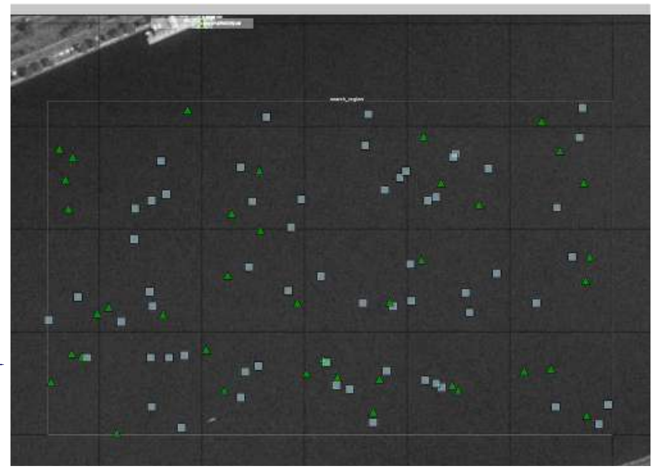
Optimization in Marine Autonomy

In **marine robotic platforms**, and real-world robots generally, decision making happens at several levels:

- Where and when is the next destination?
- What is our path plan?

- What are the sequence of heading and speed commands?

- What are the sequence of rudder and thrust commands?



Team Undergrads, 2018, MIT2.680 Hazard Search Lab

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026 — MIT Dept of Mechanical Engineering

29

MITMECHE

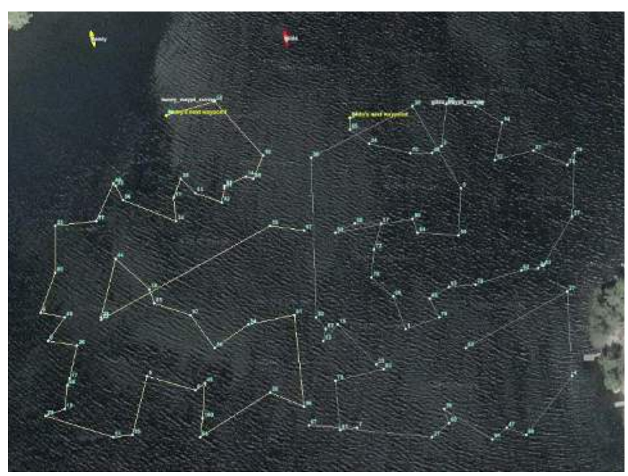
Optimization in Marine Autonomy

In **marine robotic platforms**, and real-world robots generally, decision making happens at several levels:

- Where and when is the next destination?
- What is our path plan?

- What are the sequence of heading and speed commands?

- What are the sequence of rudder and thrust commands?



MIT2.680 Distributed TSP Lab, 2018

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026 — MIT Dept of Mechanical Engineering

30

MITMECHE

Optimization in Marine Autonomy

In **marine robotic platforms**, and real-world robots generally, decision making happens at several levels:

Question:
- How much of this can be sorted out before launch?
- How much is determined on-the-fly?

• Where and when is the next destination? • What is our path plan?	Mission Autonomy	Vehicle *Agnostic* *(mostly)*
• What are the sequence of heading and speed commands?	Platform Autonomy	
• What are the sequence of rudder and thrust commands?	Platform Control	Vehicle *Dependent* *(mostly)*

31

MITMECHE

If the World Were Totally Predictable

Courtesy, Black Sheep Videos, https://vimeo.com/106226560

32

MITMECHE

If the World Were Totally Predictable

Courtesy, https://www.youtube.com/watch?v=zdOTV2RH9IY

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026

MIT Dept of Mechanical Engineering

33

MITMECHE

If the World Were Totally Predictable (but it's not)

Fake World: Total Predictability

Black Sheep Videos, https://vimeo.com/106226560

Real World: Some Predictability and a lot of improvising

Time Labse SAIL 2015: https://www.youtube.com/watch?v=ePDoCPi06rk

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026

MIT Dept of Mechanical Engineering

34

MITMECHE

Optimization in Marine Autonomy

In the Linear Programming example, the decision was simply to decide how many chairs, and how many tables to build (x_1, and x_2).

In robotic platforms, decision making happens at several levels:

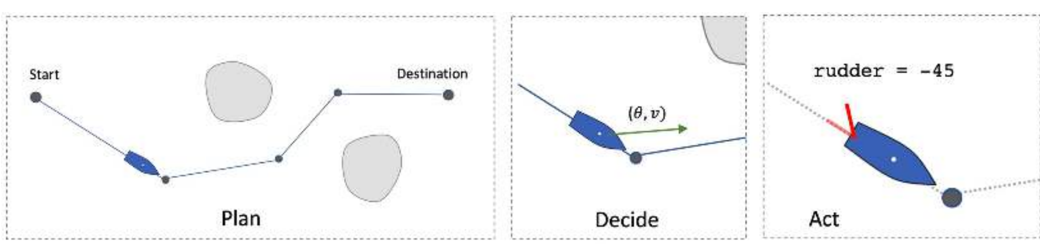
	• Where and when is the next destination? • What is our path plan?	Mission Autonomy	Vehicle *Agnostic* *(mostly)*
OUR FOCUS	• What are the sequence of heading and speed commands?	Platform Autonomy	↑
	• What are the sequence of rudder and thrust commands?	Platform Control	Vehicle *Dependent* *(mostly)*

35

MITMECHE

Optimization in Marine Autonomy

In **marine robotic platforms**, and real-world robots generally, decision making happens at several levels:



Vehicle *Agnostic* *(mostly)* ← Vehicle *Dependent* *(mostly)*

36

MITMECHE

Multi-Objective Optimization

37

MITMECHE

Multi-Objective Optimization

Concept introduction

- Multiple objective functions over the same decision space.
- Metrics are typically uncorrelated – optimizing apples vs. oranges.
- Let's return to the furniture example: It has single objective function – to maximize revenue:

Objective Function: fuction maximize $z = 20x_1 + 15x_2$ objective

subject to:

$$\begin{aligned} x_2 &\le 8 \\ 2x_1 - x_2 &\le 0 \\ 2x_1 + x_2 &\le 12.5 \\ x_1 &\ge 0 \\ x_2 &\ge 0 \end{aligned}$$

constraints

The Oakwood Furniture Company has 12.5 units of wood on hand from which to manufacture tables and chairs. Making a table uses two units of wood and making a chair uses one unit. Oakwood's distributor will pay $20 for each table and $15 for each chair, but he will not accept more than eight chairs and he wants at least twice as many chairs as tables. How many tables and chairs should the company produce to maximize revenue?

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026
MIT Dept of Mechanical Engineering

38

Multi-Objective Optimization

Concept introduction

- Multiple objective functions over the same decision space.
- Metrics are typically uncorrelated – optimizing apples vs. oranges.
- Let's return to the furniture example: It has single objective function – to maximize revenue:

Objective Function: maximize $z = 20x_1 + 15x_2$ objective fuction

subject to:

$$\begin{aligned} x_2 &\le 8 \\ 2x_1 - x_2 &\le 0 \\ 2x_1 + x_2 &\le 12.5 \\ x_1 &\ge 0 \\ x_2 &\ge 0 \end{aligned}$$

constraints

The Oakwood Furniture Company has 12.5 units of wood on hand from which to manufacture tables and chairs. Making a table uses two units of wood and making a chair uses one unit. Oakwood's distributor will pay $20 for each table and $15 for each chair, but he will not accept more than eight chairs and he wants at least twice as many chairs as tables. How many tables and chairs should the company produce to maximize revenue?

Question: What if Oakwood also wants to maximize market presence? In other words, sell as many items as possible, chairs or tables.

Multi-Objective Optimization

Concept introduction

- Multiple objective functions over the same decision space.
- Metrics are typically uncorrelated – optimizing apples vs. oranges.

Objective Function: maximize

$z = x_1 + x_2$ objective fuction #1

$z = 20x_1 + 15x_2$ objective fuction #2

subject to:

$$\begin{aligned} x_2 &\le 8 \\ 2x_1 - x_2 &\le 0 \\ 2x_1 + x_2 &\le 12.5 \\ x_1 &\ge 0 \\ x_2 &\ge 0 \end{aligned}$$

constraints

The end-goal may be to maximize revenue "overall in the future". The management may be certain that they want to both maximize profit and maximize market share, but may have no clue what the right mix may be to maximize revenue 10 years out.

Not knowing what that "right mix" may be wouldn't preclude trying to at least explore what the options may be. (Exploring the Pareto frontier).

MITMECHE

Multi-Objective Optimization
Definition

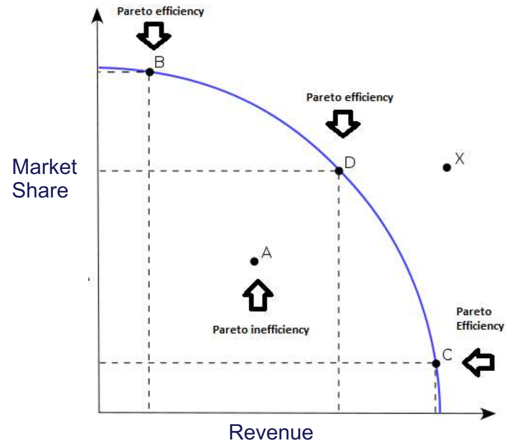
- A multi-objective optimization problems may be expressed as

$$\min_{x} \; f_1(x), f_2(x), \ldots f_n(x)$$

Typically, there is no definitive solution to this problem, but rather a family of solutions – Pareto Optimal solutions.

A Pareto optimal solution is one where improvement on one objective function cannot be achieved without sacrificing performance on another objective function.

A Pareto Optimal solution is also called a non-dominated solution.



MITMECHE

Pareto Optimality
Simple Example

Your goal after graduation is to find a job that both:
- Pays well
- Close to where your significant other lives.

Company	Salary	Distance
iRobot	$165,000	37 miles
Bluefin Robotics	$186,000	55 miles
Clearpath Robotics	$212,000	342 miles
Rethink Robotics	$182,000	45 miles
Robotic Marine Systems	$147,000	65 miles
Jaybridge Robotics	$154,000	119 miles
Boston Dynamics	$192,000	76 miles
Black-I Robotics	$184,000	122 miles
Honeybee Robotics	$139,000	144 miles
Friendly Robotics	$169,000	94 miles

MITMECHE

Pareto Optimality
Simple Example

Your goal after graduation is to find a job that both:
- Pays well
- Close to where your significant other lives.

Dominated choices

Company	Salary	Distance
iRobot	$165,000	37 miles
Bluefin Robotics	$186,000	55 miles
Clearpath Robotics	$202,000	342 miles
Rethink Robotics	$182,000	55 miles
Robotic Marine Systems	$147,000	65 miles
Jaybridge Robotics	$154,000	119 miles
Boston Dynamics	$192,000	76 miles
Black-I Robotics	$184,000	122 miles
Honeybee Robotics	$139,000	144 miles
Friendly Robotics	$169,000	94 miles

Dominates:

Dominates:

43

MITMECHE

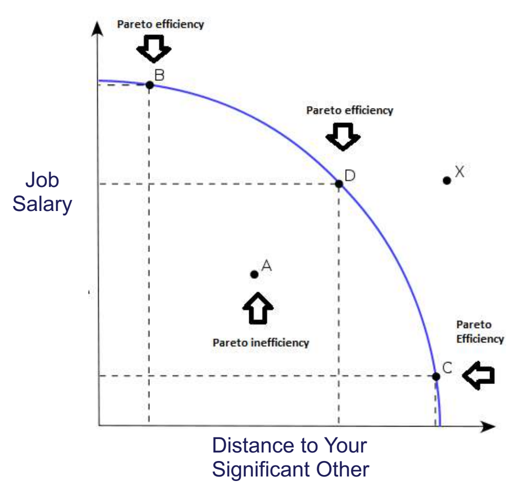
Multi-Objective Optimization
Definition

- A multi-objective optimization problems may be expressed as

$$\min_{x} \; f_1(x), f_2(x), \ldots f_n(x)$$

The term value function is sometimes used to refer to the decision-makers relative preference in optimizing each objective function.

The user may not precisely know their own value function but may come to discover it by exploring tradeoffs in the Pareto Optimal space. (A good visualization GUI helps.)



44

Solutions that perform well in multiple criteria are often implied to be Pareto Optimal

45

MITMECHE

Solutions that perform well in multiple criteria are often implied to be Pareto Optimal

This car may not even be Pareto Optimal w.r.t."Great ride" and "Gets great mileage"!!

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools
Michael Benjamin, ©2026
MIT Dept of Mechanical Engineering

46

MITMECHE

Solutions that perform well in multiple criteria are often implied to be Pareto Optimal

This car may not even be Pareto Optimal w.r.t. “Voice Activated” and “Gets great mileage”!!

MITMECHE

Pareto Optimal solutions are conflated with being optimal across all value functions

MITMECHE

Pareto Optimal solutions are conflated with being optimal across all value functions

	Torque	Mileage
Ford	398 ft-lbs	14.7
Other	380 ft-lbs	23.1

- Claims to be Pareto Optimal (and maybe it is!)
- It may just dominate all other trucks on the issue of “torque”, and be worst on gas mileage

MITMECHE

What is the role of Multi-Objective Optimization on a Robot?

What is the role of Multi-Objective Optimization on a Robot?

- The decision-maker is the robot.
- It has to have a clear relative preference between objective functions (value function)
- It needs to make a decision and move on!

Multi-Objective Optimization On a Robot

- The robot multi-objective optimization problems may be expressed as

$$\min_{x} \; f_1(x), f_2(x), \ldots f_n(x)$$

How does the robot automatically solve a multi-objective optimization problem.
By automatically we mean, automatically determine the value function (relative importance of functions)

If there were a human in the loop, they could explore different value functions, e.g., trading off distance travelled, vs. safety etc.

But there is no human in the loop, and the decisions are happening several times per second.
The robot must have a value function – relative weight of its multiple objective functions.

MITMECHE

Methods for Setting a Value Function

There are a few different ways to automatically set the value function in a multi-objective optimization problem.

- Pick the most important objective function and ignore the rest.
  Seems draconian but on a robot, this may make sense in some applications.
- "Good enough" search. Define a level of performance for each objective function that is good enough. Optimization of one objective function is done over a set of decisions deemed to satisfy the good enough criteria of preceding objective funtions.
- Constructing a single aggregate objective function.
  Each objective function is given a priority weight.

MITMECHE

Multi-Objective Optimization in the IvP Helm

Behavior 1 → $f_1(x_1, x_2, \ldots, x_n)$

Behavior 2 → $f_2(x_1, x_2, \ldots, x_n)$

Behavior 3 → $f_3(x_1, x_2, \ldots, x_n)$

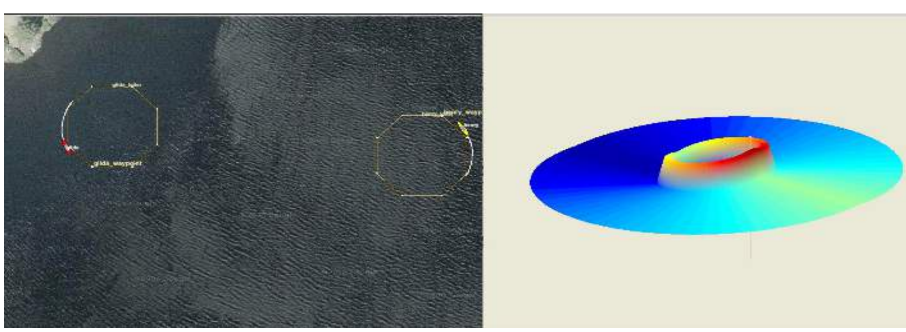
Objective Functions

Solver

$$\vec{x}^* = \underset{\vec{x}}{\operatorname{argmax}} \sum_{i=0}^{k-1} w_i f_i(\vec{x})$$

→ Action

- The solution, $\vec{x}^*$, is the single decision maximizing the weighted sum of all utility functions.

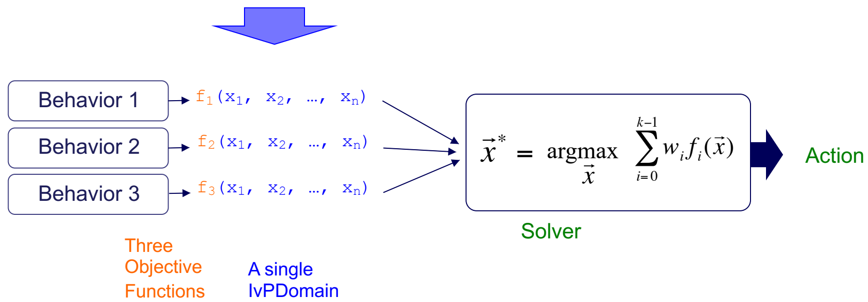


MITMECHE

The IvPDomain

Overview

The IvPDomain is a data structure representing the decision space common to all objective functions produced by helm behaviors:



MITMECHE

The IvPDomain

Discrete vs. Continuous

The IvPDomain is a discrete domain.

- There are a finite number of possible decisions.
- The possible decisions are uniformly spaced, e.g., speed = {0, 0.5, 1.0, 1.5, 2.0}
- Brute force (exhaustive enumeration) in theory is an option, in practice too slow.
- A solution algorithm that explictly or implicitly considers all decisions may be considered globally optimal.

MITMECHE

The IvPDomain

Discrete vs. Continuous

The domain is discrete since control over the vehicle actuators may only have limited precision.

- Fruitless to reason about a heading=45.0024 if the vehicle's heading sensor only provides precision to the nearest degree.
- Even if the sensor were able to provide this information, there may be limited difference in utility between 45.1 and 45.2 degrees.
- Vehicle heading is reconsidered on each iteration. So even though a heading of 45.72 degrees may be needed to reach a waypoint a kilometer away, a series of heading commands fluctuating between 45 and 46 can achieve this goal.

57

MITMECHE

The IvPDomain

Defining the domain in the mission file

The IvPDomain is defined in the pHelmIvP configuration block in the mission.moos configuration file:

```
ProcessConfig = pHelmIvP
{
  AppTick    = 4
  CommsTick  = 4

  Behaviors  = charlie.bhv
  Verbose    = true
  Domain     = course:0:359:360
  Domain     = speed:0:4:21
  Domain     = depth:0:490:491
}
```

3,711,960 possible decisions

The above domain has three decision variables, course, speed, and depth.

- The course variable has 360 choices ranging from 0 to 359 degrees.
- The speed variable has 21 choices ranging from 0 to 4 meters per sec.
- The depth variable has 491 choices ranging from 0 to 490 meters.

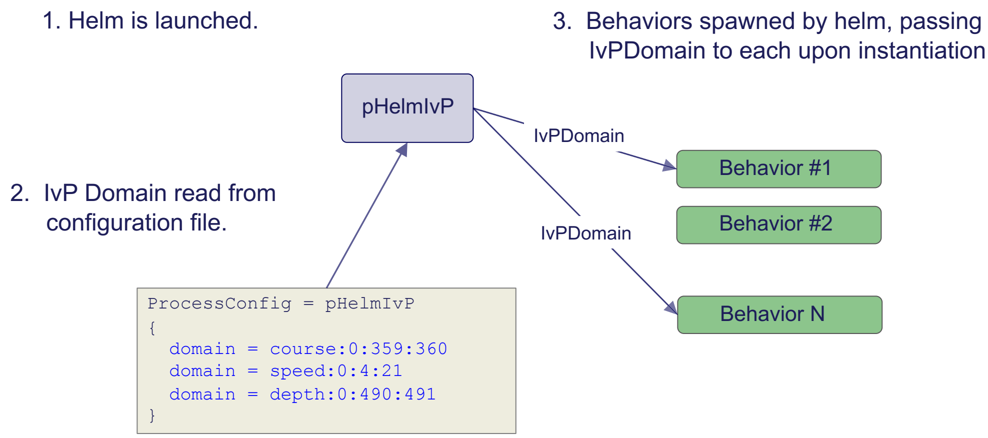
All helm behaviors must reason over one or all of these three variables.

58

MITMECHE

Helm Domain Handling

Events upon helm start-up



MITMECHE

The IvPDomain

Receiving and Refining in the Constructor

The Constructor:

```
01  // Behavior Constructor
02  BHV_SimpleWaypoint::BHV_SimpleWaypoint(IvPDomain domain) :
03                      IvPBehavior(domain)
04  {
05    m_domain = subDomain(m_domain, "course,speed");
06    addInfoVars("NAV_X, NAV_Y");
07  }
```

Lines 2-3

- The domain is passed to the behavior upon instantiation by the helm.
- The domain is known to the helm from the helm configuration block.
- The domain is handled by the IvPBehavior superclass constructor.
- The result is that the m_domain member variable reflects the domain.

Line 5

- The domain is refined down to include only course and speed variables.
- This objective function produced by this behavior will be defined only over course and speed.

MITMECHE

IvP Sub-Domains

Definition and example

An IvPDomain A is a sub-domain of another IvPDomain B, if

- the set of decision variables in A is a subset of the variables in B, and
- the set of decisions for each variable is the same for each domain.

A
```
ProcessConfig = pHelmIvP
{
  Domain = course:0:359:360
  Domain = speed:0:4:21
  Domain = depth:0:490:491
}
```

B
```
ProcessConfig = pHelmIvP
{
  Domain = course:0:359:360
  Domain = speed:0:4:21
  Domain = depth:0:490:491
}
```

Is B a sub-domain of A? Yes

61

MITMECHE

IvP Sub-Domains

Definition and example

A IvPDomain A is a sub-domain of another IvPDomain B, if

- the set of decision variables in A is a subset of the variables in B, and
- the set of decisions for each variable is the same for each domain.

A
```
ProcessConfig = pHelmIvP
{
  Domain = course:0:359:360
  Domain = speed:0:4:21
  Domain = depth:0:490:491
}
```

B
```
ProcessConfig = pHelmIvP
{
  Domain = course:0:359:360
  Domain = speed:0:4:21
}
```

Is B a sub-domain of A? Yes

62

IvP Sub-Domains

Definition and example

A IvPDomain A is a sub-domain of another IvPDomain B, if

- the set of decision variables in A is a subset of the variables in B, and
- the set of decisions for each variable is the same for each domain.

A

```
ProcessConfig = pHelmIvP
{
  Domain = course:0:359:360
  Domain = speed:0:4:21
  Domain = depth:0:490:491
}
```

B

```
ProcessConfig = pHelmIvP
{
  Domain = course:0:359:360
  Domain = speed:0:4:5
}
```

Is B a sub-domain of A?

63

MITMECHE

IvP Sub-Domains

Definition and example

A IvPDomain A is a sub-domain of another IvPDomain B, if

- the set of decision variables in A is a subset of the variables in B, and
- the set of decisions for each variable is the same for each domain.

A

```
ProcessConfig = pHelmIvP
{
  Domain = course:0:359:360
  Domain = speed:0:4:21
  Domain = depth:0:490:491
}
```

B

```
ProcessConfig = pHelmIvP
{
  Domain = course:0:359:360
  Domain = speed:0:4:5
}
```

Is B a sub-domain of A? NO

64

IvP Sub-Domains

The subDomain() Utility Function

The subDomain() utility function may be used to create a proper subdomain of another given IvPDomain:

Defined in `lib_ivpbuild/BuildUtils.h`

```
IvPDomain  subDomain(IvPDomain, string);
```

```
// Behavior Constructor
BHV_SimpleWaypoint::BHV_SimpleWaypoint(IvPDomain domain) :    \
                    IvPBehavior(domain)
{
  m_domain = subDomain(m_domain, "course,speed");
  addInfoVars("NAV_X, NAV_Y");
}
```


MITMECHE

IvP Sub-Domains

The subDomain() Utility Function

If the domain variables specified in the subDomain() call are not in the given domain, the returned domain will be empty.

```
m_domain = subDomain(m_domain, "course,speed");

if(m_domain.size() == 0)
    return(false);
```

MITMECHE

IvP Domains
and
IvP Functions

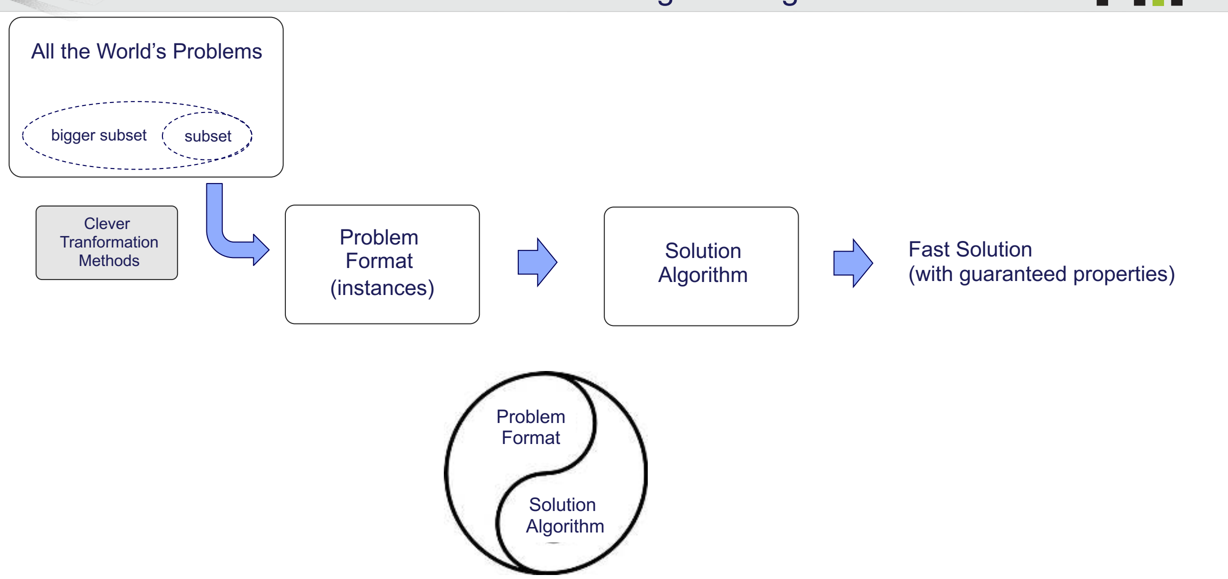
An IvP Function is defined over an IvP Domain.

What is an IvP Function?

67

MITMECHE

Mathematical Programming



68

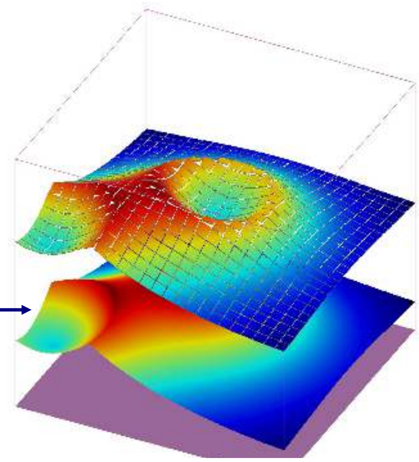
MITMECHE

IvP Functions

The IvP Function vs. Underlying Function

An *IvP function* is a piecewise linear approximation of an objective function, over a discrete decision space (domain).

Piecewise Linear Approximation 525 Pieces



Underlying Function

$$f_i(x,y) = ((1 - \frac{|\sqrt{(x-250)^2+(y-250)^2} - 100|}{2500})^8 * 200) - 100, \mid ((1 - \frac{|\sqrt{(x-50)^2+(y-50)^2} - 100|}{2500})^8 * 200) - 100$$

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | **IvP Functions** | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026 MIT Dept of Mechanical Engineering

69

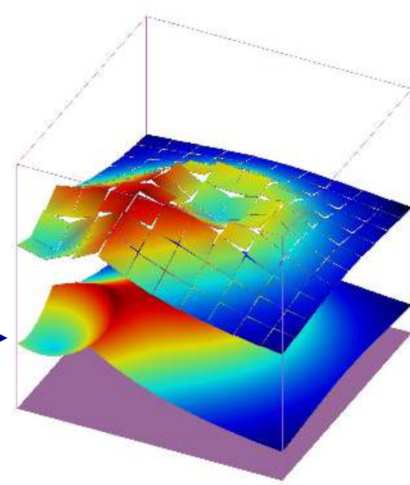
MITMECHE

IvP Functions

The IvP Function vs. Underlying Function

An *IvP function* is a piecewise linear approximation of an objective function, over a discrete decision space (domain).

Piecewise Linear Approximation 100 Pieces



Underlying Function

$$f_i(x,y) = ((1 - \frac{|\sqrt{(x-250)^2+(y-250)^2} - 100|}{2500})^8 * 200) - 100, \mid ((1 - \frac{|\sqrt{(x-50)^2+(y-50)^2} - 100|}{2500})^8 * 200) - 100$$

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | **IvP Functions** | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026 MIT Dept of Mechanical Engineering

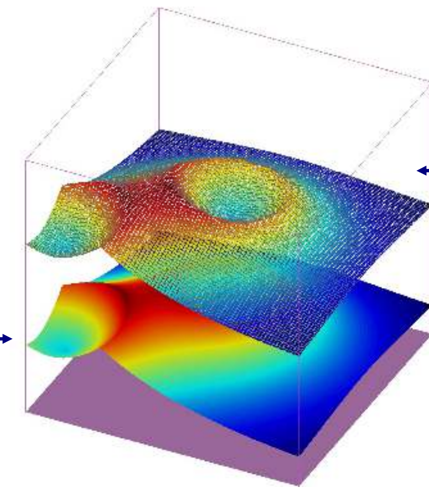
70

MITMECHE

IvP Functions

The IvP Function vs. Underlying Function

An *IvP function* is a piecewise linear approximation of an objective function, over a discrete decision space (domain).



$$f_i(x,y) = ((1-\frac{|\sqrt{(x-250)^2+(y-250)^2}-100|}{2500})^8 * 200) - 100, \; ((1-\frac{|\sqrt{(x-50)^2+(y-50)^2}-100|}{2500})^8 * 200) - 100$$

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

MIT Dept of Mechanical Engineering

71

MITMECHE

IvP Functions

Piece Format and Properties

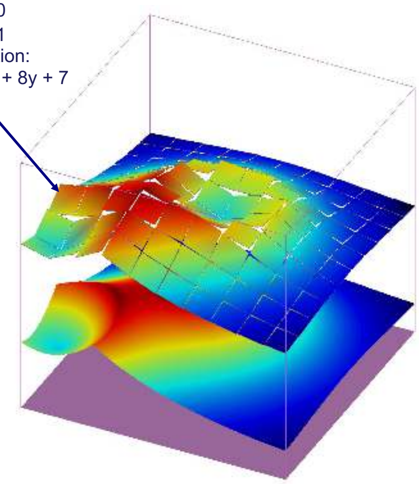
Piecewise linear (IvP) functions:

- Each point in the decision space is contained by exactly one piece
- Each pieces has an interval boundary and a linear interior function.

Limitations:

- A piecewise linear function is only an *approximation* of the underlying function.
- But - the user has discretion over the number of pieces, distribution of pieces and time used to create the approximation.

Interval Boundary:
$10 \le x \le 20$
$14 \le y \le 21$
Interior Function:
$f(x,y) = 4x + 8y + 7$



Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

MIT Dept of Mechanical Engineering

72

MITMECHE

IvP Functions

Piece Intersection

Piece Intersection:

- In any one given IvP function, no two pieces intersect (overlap).
- In the IvP solution algorithm (over multiple functions) piece intersection is a key idea.

p_a

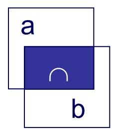
Boundary:
$10 \le x \le 20$
$14 \le y \le 21$
Interior Function:
$f(x,y) = 4x + 8y + 7$

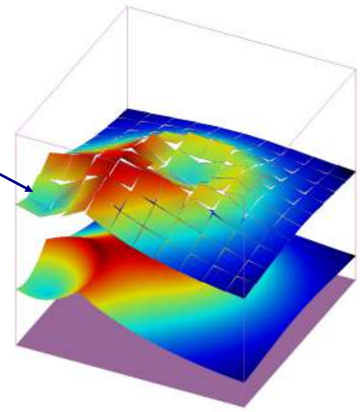
p_b

Boundary:
$15 \le x \le 25$
$21 \le y \le 30$
Interior Function:
$f(x,y) = 2x + 5y + 2$

$p_a \cap p_b$

Boundary:
$15 \le x \le 20$
$21 \le y \le 21$
Interior Function:
$f(x,y) = 6x + 13y + 9$





73

MITMECHE

IvP Functions and IvP Behaviors

The primary output of a IvPBehavior is an IvPFunction, in the onRunState() function.

```
IvPFunction *BHV_YourBehavior::onRunState()
{
  IvPFunction *ipf = generateIvPFunction()
  return(ipf);
}
```

The IvPBuild Toolbox (lib_ivpbuild) contains a number of "build tools" to facilitate the production of IvP Functions.

We explore this next.

74

MITMECHE

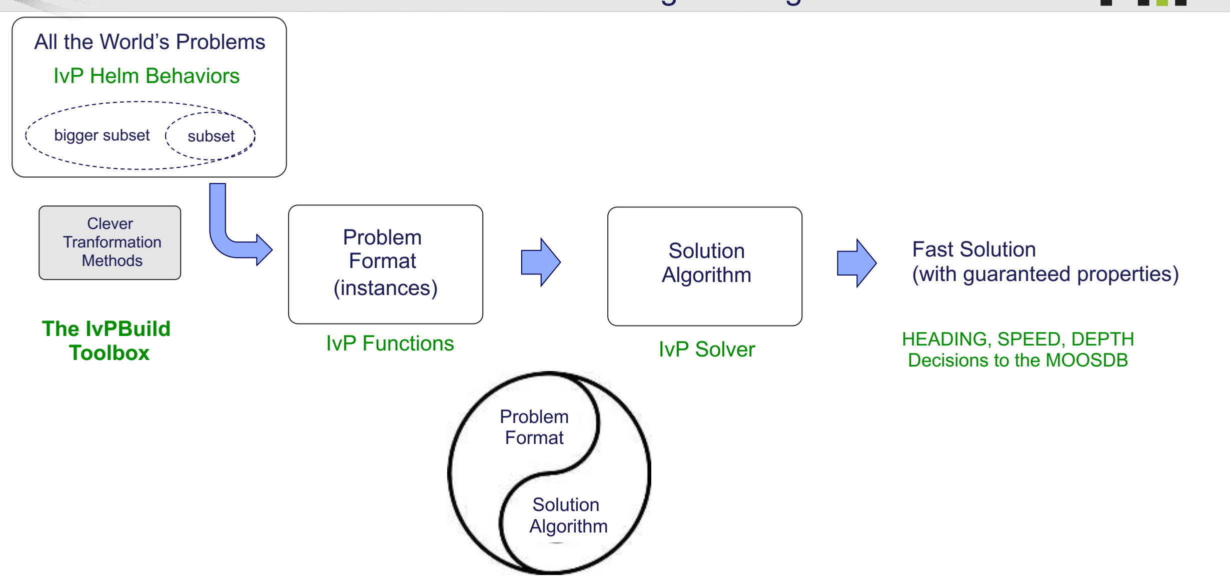
The IvP Build Toolbox

A set of utilities for building IvP functions

75

MITMECHE

Mathematical Programming



76

MITMECHE

The IvPBuild Toolbox

General Usage Pattern

The IvPBuild Toolbox: a set of tools to facilitate building IvPFunctions.
The tools each do different things, but all work in the same general way:

1. Create a BuildTool instance. — Build Tool
2. Pass the IvPDomain to the tool. — IvPDomain → Build Tool
3. Pass Parameters to the tool. — Params → Build Tool
4. Extract the IvP Function. — IvPFunction ← Build Tool

MITMECHE

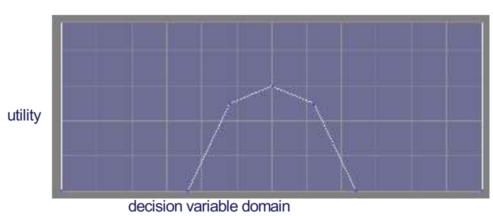
IvP Build Toolbox

Two Categories of Tools

The IvP Build Tools have two general categories:

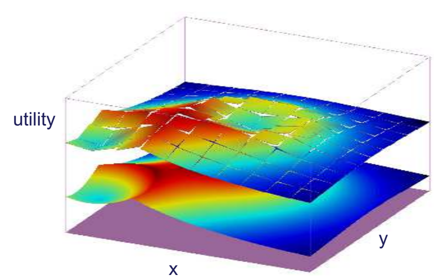
The ZAIC Toolset

Building 1 dimension objective functions
(functions over a single variable)



The Reflector Toolset

Building N dimension objective functions
(functions over a multiple variables)



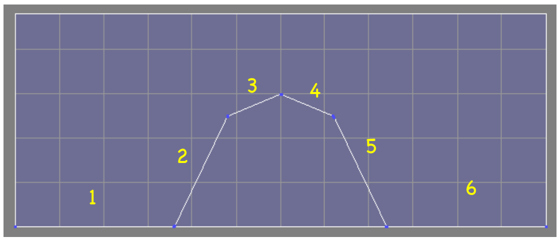
MITMECHE

The ZAIC Build Tools

IvP Functions over a single variable

The ZAIC build tools are a family of tools for generating objective functions with a single decision variable.

utility



decision variable domain

There are four tools in this set:

1. ZAIC_PEAK
2. ZAIC_LEQ
3. ZAIC_HEQ
4. ZAIC_Vector

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026

MIT Dept of Mechanical Engineering

79

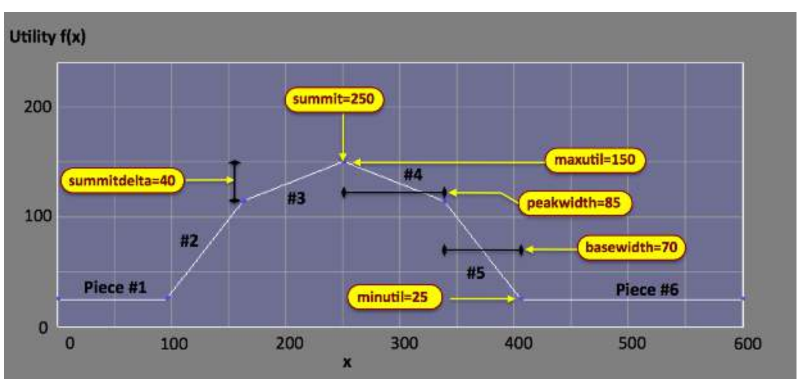
MITMECHE

The ZAIC_Peak Tool

The ZAIC_Peak tool is designed with the objective function form below in mind.

- A preferred decision (the summit), with maximum utility (maxutil).
- A drop-off in utility as the variable value deviates from the preferred choice.

utility



decision variable domain

Optimization Problems | Multi-Objective Optimization | Robot IvP Helm Optimization | The IvP Domain | IvP Functions | IvPBuild ZAIC Tools | IvPBuild Reflector Tools

Michael Benjamin, ©2026

MIT Dept of Mechanical Engineering

80

The ZAIC Build Tools

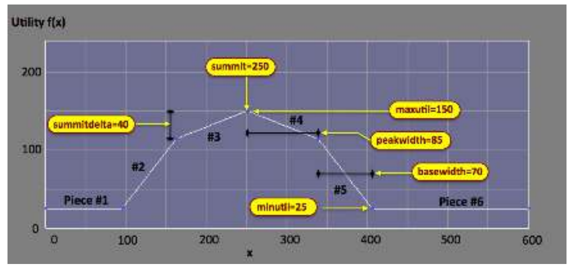
The ZAIC_Peak Tool

Typical usage of the tool in code:

```
01 ZAIC_PEAK zaic_peak(domain, "depth");
02
03 zaic_peak.setSummit(150);
04 zaic_peak.setMinMaxUtil(20, 120);
05 zaic_peak.setBaseWidth(60);
06
07 IvPFunction *ipf = 0;
08 ipf = zaic_peak.extractIvPFunction();
```

01	Create the ZAIC instance, passing the overall IvPDomain and particular variable.
03-05	Set the desired ZAIC parameters.
07-08	Extracting the IvPFunction from the ZAIC tool.

utility



decision variable domain

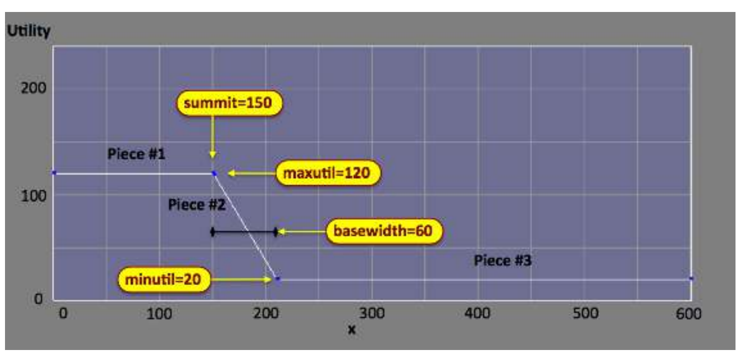
The ZAIC Build Tools

The ZAIC_LEQ Tool

The ZAIC_LEQ tool is designed with the objective function form below in mind.

- The summit parameter is the point where max utility begins to drop off.
- The minutil parameter has default 0. The maxutil parameter has default 100.
- The basewidth parameter may be used to soften the drop in utility.

utility



decision variable domain

The ZAIC_Peak Tools

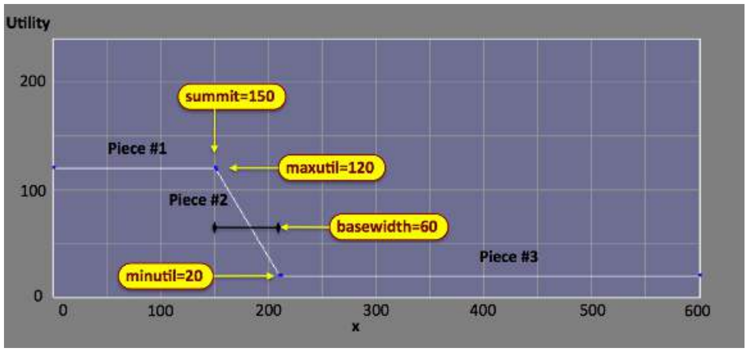
- When basewidth=0

$$f(x) = \begin{cases} \texttt{maxutil} & x \le \texttt{summit}, \\ \texttt{minutil} & \text{otherwise.} \end{cases}$$

- When basewidth != 0

$$f(x) = \begin{cases} \texttt{maxutil} & x \le \texttt{summit}, \\ \texttt{minutil} + ((\texttt{maxutil - minutil}) * ((\texttt{x - summit}) \ / \ \texttt{basewidth})) & \texttt{summit} < x \le \texttt{summit} + \texttt{basewidth}, \\ \texttt{minutil} & \text{otherwise.} \end{cases}$$

utility

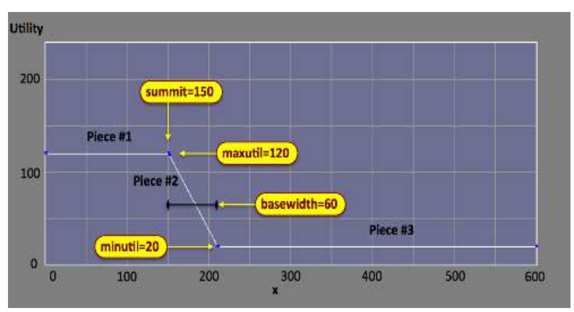


decision variable domain

The ZAIC_LEQ Tool

Typical code structure:

```
01  ZAIC_LEQ zaic(domain, "depth");
02
03  zaic.setSummit(150);
04  zaic.setBaseWidth(60);
05
06  IvPFunction *ipf = 0;
07  ipf = zaic.extractIvPFunction();
```



decision variable domain

01	Create the ZAIC instance, the overall IvPDoman and particular variable.
03-04	Set the desired ZAIC parameters.
06-07	Extracting the IvPFunction from the ZAIC tool.

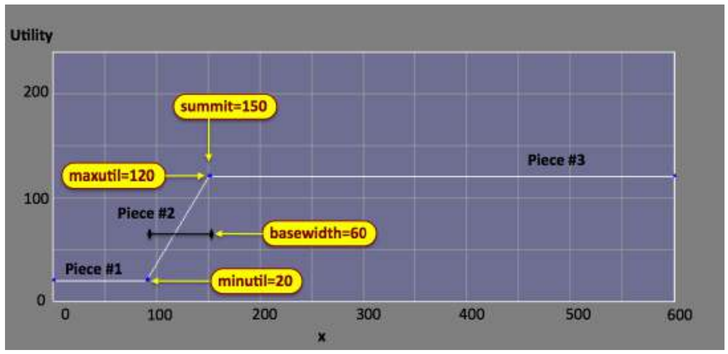
MITMECHE

The ZAIC_HEQ Tool

The ZAIC_HEQ tool is designed with the objective function form below in mind.

- The summit parameter is the point where max utility begins to drop off.
- The minutil parameter has default 0. The maxutil parameter has default 100.
- The basewidth parameter may be used to soften the drop in utility.

utility



decision variable domain

85

MITMECHE

The ZAIC_HEQ Tool

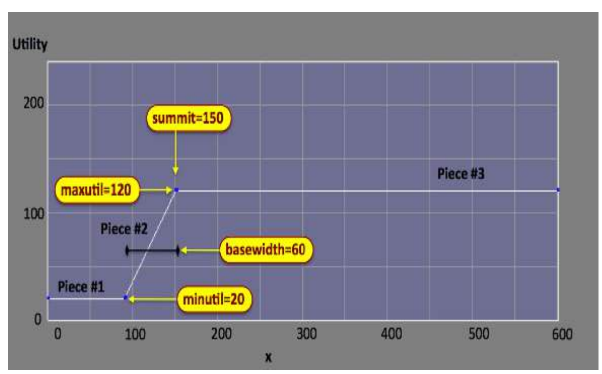
Typical code structure:

```
01  ZAIC_HEQ zaic(domain, "depth");
02
03  zaic.setSummit(150);
04  zaic.setBaseWidth(60);
05
06  IvPFunction *ipf = 0;
07  ipf = zaic.extractIvPFunction();
```

01 Create the ZAIC instance, the overall IvPDoman and particular variable.

03-04 Set the desired ZAIC parameters.

06-07 Extracting the IvPFunction from the ZAIC tool.



decision variable domain

86

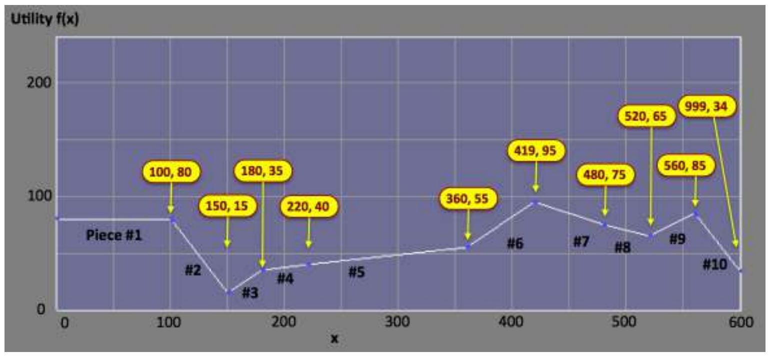
MITMECHE

The ZAIC_Vector Tool

The ZAIC_Vector Tool is a catch-all tool for one-variable objective functions

- It accepts two equally sized vectors of numerical values (doubles).
- A vector of domain values.
- A vector of utility values.

utility



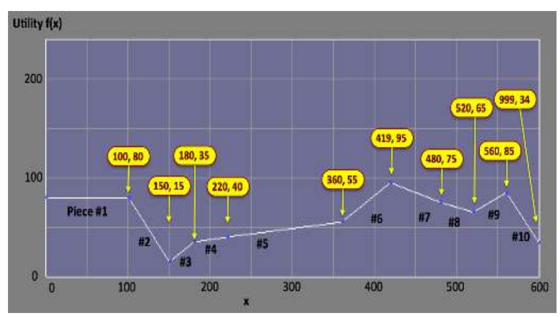
decision variable domain

MITMECHE

The ZAIC_Vector Tool

Typical code structure:

```
01 ZAIC_Vector zaic(domain, "depth");
02
03 vector<double> domain;
04 vector<double> range;
05 domain.push_back(100); range.push_back(80);
06 domain.push_back(150); range.push_back(15);
07 domain.push_back(180); range.push_back(35);
08
09 zaic.setDomainVals(domain);
10 zaic.setRangeVals(range);
11
12 IvPFunction *ipf = 0;
13 ipf = zaic.extractIvPFunction();
```



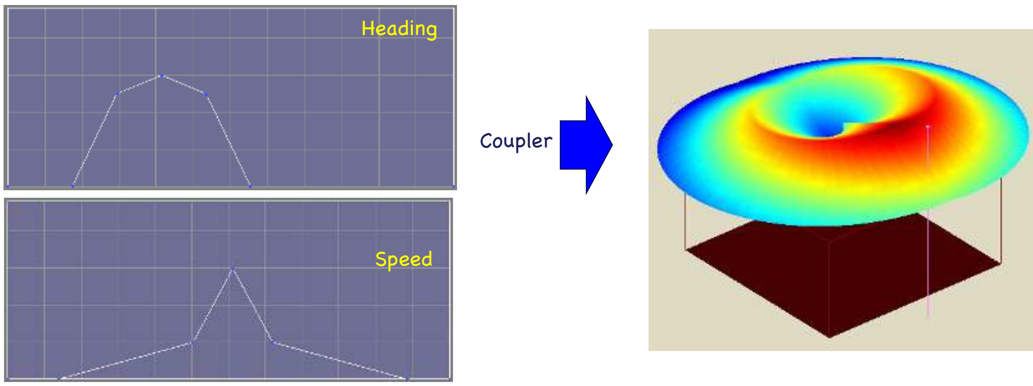
decision variable domain

01	Create the ZAIC instance, the overall IvPDoman and particular variable.
03-07	Build the pair of vectors.
09-10	Pass the vectors to the ZAIC tool.
06-07	Extracting the IvPFunction from the ZAIC tool.

The OF_Coupler Tool

The OF_Coupler tool is used to combine two objective functions into a single, combined function.

- The two objective functions must be defined over different variables.
- The resulting function will be defined over the coupled domain.



The OF_Coupler Tool

Typical usage of the coupler in code:

```
01 IvPFunction *ipf_1 = zaic_1.extractIvPFunction();
02 IvPFunction *ipf_2 = zaic_2.extractIvPFunction();
03
04 OF_Coupler coupler;
05 IvPFunction *ipf = coupler.couple(ipf_1, ipf_2, 50, 50);
```

01 The first IvP Function is created.

02 The second IvP Function is created.

04 A OF_Coupler tool is created.

04 The new coupled IvP Function is generated.

MITMECHE

The OF_Coupler Tool

Typical usage of the coupler in code:

```
01 IvPFunction *ipf_1 = zaic_1.extractIvPFunction();
02 IvPFunction *ipf_2 = zaic_2.extractIvPFunction();
03
04 OF_Coupler coupler;
05 IvPFunction *ipf = coupler.couple(ipf_1, ipf_2, 50, 50);
```

- 01 The first IvP Function is created.
- 02 The second IvP Function is created.
- 04 A OF_Coupler tool is created.
- 04 The new coupled IvP Function is generated.

Note: the weights (50, 50) reflect the relative contribution of each function to the coupled function.

91

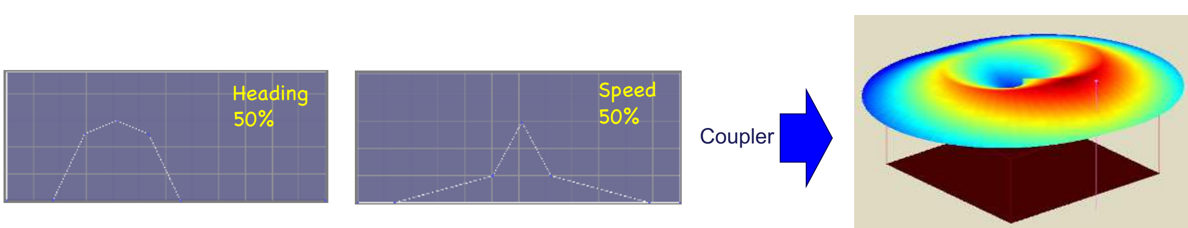
MITMECHE

The OF_Coupler Tool

Typical usage of the coupler in code:

```
01 IvPFunction *ipf_1 = zaic_1.extractIvPFunction();
02 IvPFunction *ipf_2 = zaic_2.extractIvPFunction();
03
04 OF_Coupler coupler;
05 IvPFunction *ipf = coupler.couple(ipf_1, ipf_2, 50, 50);
```

Note: the weights (50, 50) reflect the relative contribution of each function to the coupled function.



92

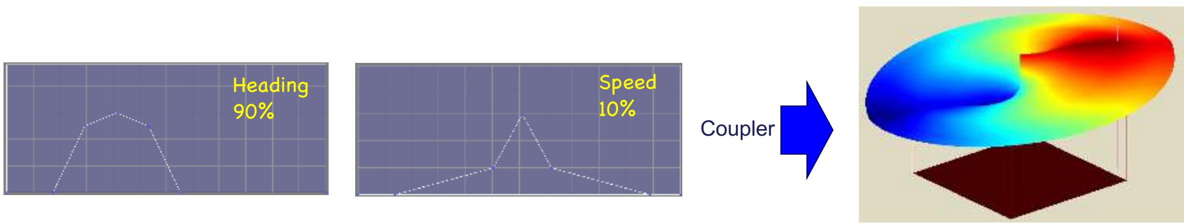
The OF_Coupler Tool

Typical usage of the coupler in code:

```
IvPFunction *ipf_1 = zaic_1.extractIvPFunction();
IvPFunction *ipf_2 = zaic_2.extractIvPFunction();

OF_Coupler coupler;
IvPFunction *ipf = coupler.couple(ipf_1, ipf_2, 50, 50);
```

Note: the weights (50, 50) reflect the relative contribution of each function to the coupled function.



MIT Dept of Mechanical Engineering

93

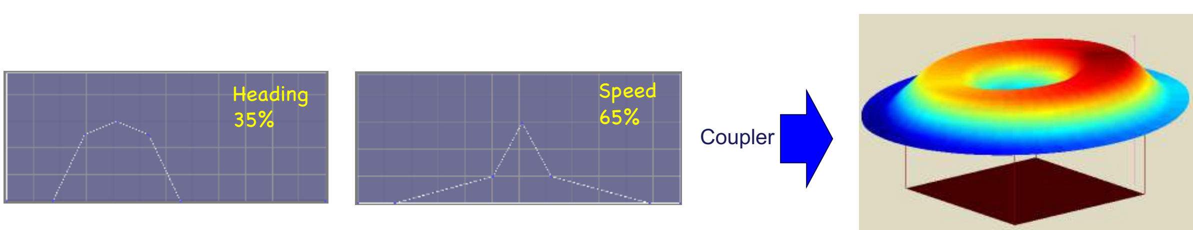
The OF_Coupler Tool

Typical usage of the coupler in code:

```
IvPFunction *ipf_1 = zaic_1.extractIvPFunction();
IvPFunction *ipf_2 = zaic_2.extractIvPFunction();

OF_Coupler coupler;
IvPFunction *ipf = coupler.couple(ipf_1, ipf_2, 50, 50);
```

Note: the weights (50, 50) reflect the relative contribution of each function to the coupled function.

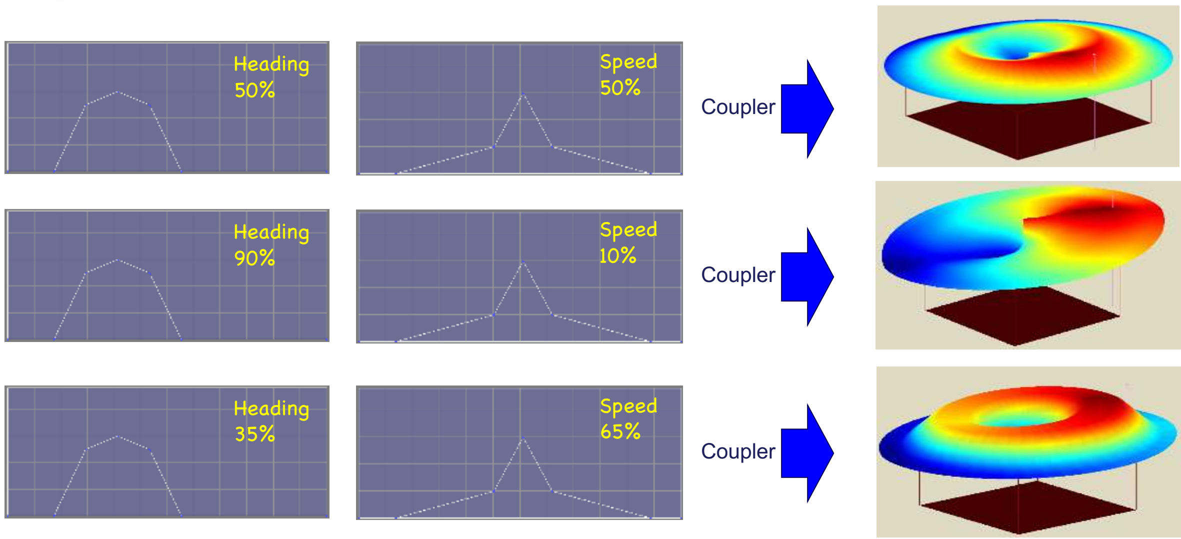


MIT Dept of Mechanical Engineering

94

MITMECHE

A Side By-Side Look



MITMECHE

Reflector Tools

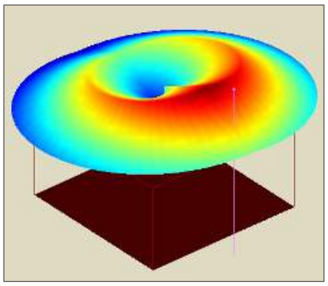
For Objective Functions with Multiple Dependent Variables

The Reflector Tool

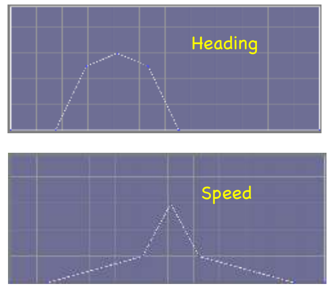
The Reflector Tool creates IvP Functions over multiple coupled decision variables.

Question: What are coupled decision variables?

In the below 2D objective function, the merits of the heading decision may be evaluated without also simultaneously considering the speed decision.…



… because the function was built by joining the two independent (decoupled) functions:



MITMECHE

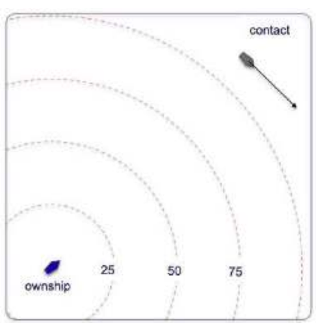
The Reflector Tool

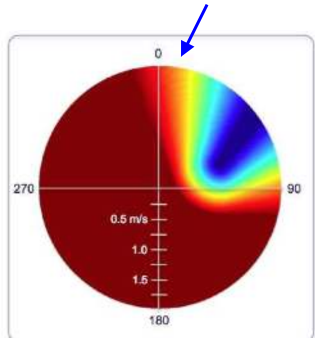
The Reflector Tool creates IvP Functions over multiple coupled decision variables.

Question: What are coupled decision variables?

In the below collision avoidance objective function, the merits of the heading decision may NOT be evaluated without also simultaneously considering the speed decision.

The below IvP Function is created with the Reflector Tool.





The IvPBuild Toolbox

General Usage Pattern

Recall The IvPBuild Toolbox Pipeline:

1. Create a BuildTool instance. — Build Tool
2. Pass IvPDomain to the tool. — IvPDomain → Build Tool
3. Pass Parameters to the tool. — Params → Build Tool
4. Extract the IvP Function. — IvPFunction ← Build Tool

99

The IvPBuild Toolbox

General Usage Pattern

Recall The IvPBuild Toolbox Pipeline:

1. Create a BuildTool instance. — Build Tool
2. Pass IvPDomain to the tool. — IvPDomain → Build Tool
3. Pass Parameters to the tool. — Params → Build Tool
4. Extract the IvP Function. — IvPFunction ← Build Tool

```
01 ZAIC_PEAK zaic_peak(domain, "depth");
02
03 zaic_peak.setSummit(150);
04 zaic_peak.setMinMaxUtil(20, 120);
05 zaic_peak.setBaseWidth(60);
06
07 IvPFunction *ipf = 0;
08 ipf = zaic_peak.extractIvPFunction();
```

- 01 — Create the ZAIC instance, passing the overall IvPDoman and particular variable.
- 03-05 — Set the desired ZAIC parameters.
- 07-08 — Extracting the IvPFunction from the ZAIC tool.

100

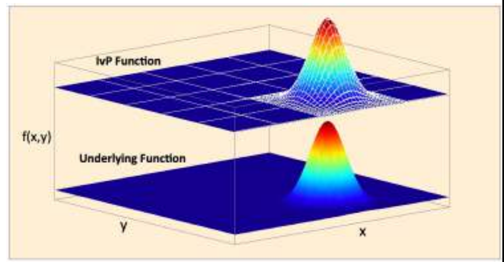
MITMECHE

The IvPBuild Toolbox

Using the Reflector Tool

With the reflector tool, the build pipeline has the additional step of passing a pointer to the underlying function to be approximated.

1. Create a BuildTool instance. → Build Tool
2. Pass IvPDomain to the tool. IvPDomain → Build Tool
3. Pass underlying function. Underlying Function → Build Tool (**New Step**)
4. Pass Parameters to the tool. Params → Build Tool
5. Extract the IvP Function. IvPFunction ← Build Tool



101

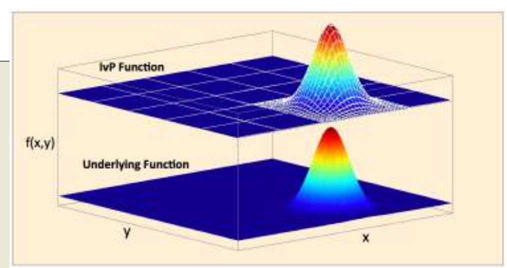
MITMECHE

The Reflector Tool

Example Code Usage

A typical code structure (usually found in the implementation of an IvP Behavior)

```
01 AOF_Gaussian aof(ivp_domain);
02 aof.setParam("xcent", 50);
03 aof.setParam("ycent", -150);
04 aof.setParam("sigma", 32.4);
05 aof.setParam("range", 150);
06
07 OF_Reflector reflector(aof);
08
09 int pieces_created = reflector.create(1000);
10 IvPFunction *ipf = reflector.extractIvPFunction();
```



- 01 Create an underlying objective function given an IvP Domain
- 02-05 Parameterize the underlying function
- 07 Create a reflector
- 09 Direct the reflector to create an approximation with 1000 pieces
- 10 Extract the objective function

102

MITMECHE

The Reflector Tool

Pure Uniform Functions

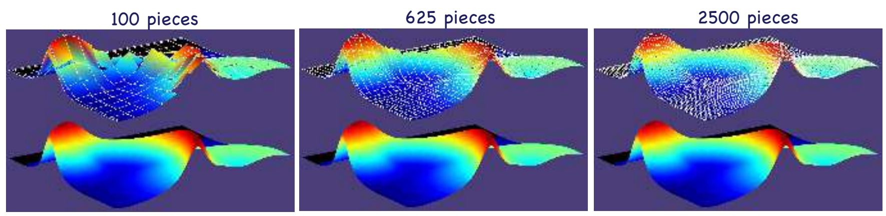
The default usage of the Reflector is to specify a desired number of pieces.

```
int pieces_created = reflector.create(1000);
```

It will return the actual number of pieces generated.

The number of pieces specified by the caller depends on:

- Accuracy desired.
- Time budget for creating the objective function.



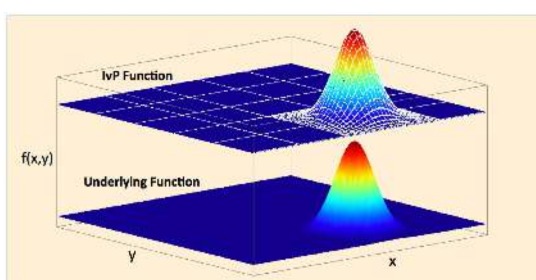
MITMECHE

The Reflector Tool

Pros and Cons of Pure Uniform Functions

- The appeal of pure uniform function generation is that it is easy to use.
- No insight needed regarding the underlying function form.

- The drawback is that it is potentially very inefficient.
- Some areas of the function domain may be very well approximated with very few pieces.



MITMECHE

The Reflector Tool

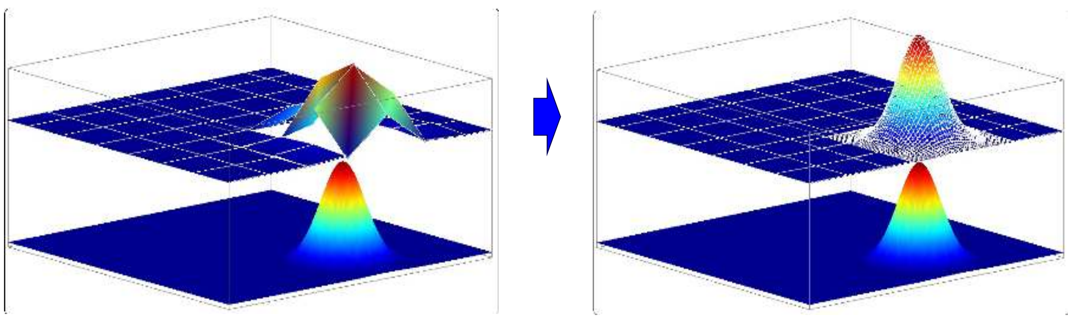
Directed Refinement

- The Directed Refinement option of the Reflector Tool allows the caller to refine a given area of the domain with a given uniform pieces size.

Basic idea:

Make an initial uniform function

Identify a sub-area of the domain.
Apply a smaller uniform piece to this area



MITMECHE

The Reflector Tool

Directed Refinement

- Typical code structure using the Directed Refinement option.

```
01 OF_Reflector reflector(aof);
02
03 reflector.setParam("uniform_piece", "discrete @ x:5,y:5");
04 reflector.setParam("refine_region", "native @ x:10:24,y:-25:20");
05 reflector.setParam("refine_piece", "discrete @ x:2,y:2");
06 reflector.create();
07
08 IvPFunction *ipf = reflector.extractIvPFunction();
```

01	Create a reflector passing it the underlying function
03	Specify the initial uniform pieces size
04	Specify the sub-region to refine
05	Specify the piece dimensions used in the refine region
06	Invoke the algorithm
08	Extract the objective function

MITMECHE

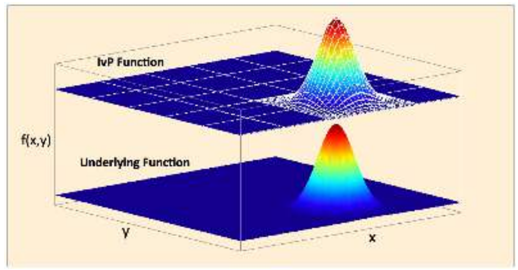
The Reflector Tool

Pros and Cons of Directed Refinement

- The appeal of directed refinement is that it is very efficient in its use of pieces and sampling of the underlying function.
- Less pieces, greater accuracy over pure uniform functions.

- The drawback is that it requires the user to actually have insight into the form of the underlying function.
- Sometimes this is the case, often it is not.

- There is another option:
  Smart Refinement



MITMECHE

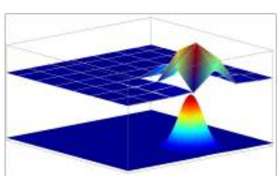
The Reflector Tool

Smart Refinement

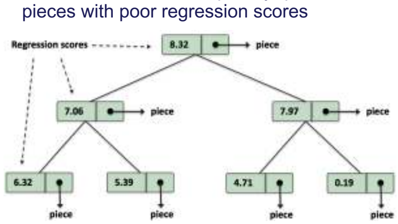
- The Smart Refinement option of the Reflector Tool allows the caller to automatically identify pieces that may need further refinement.

Basic idea:

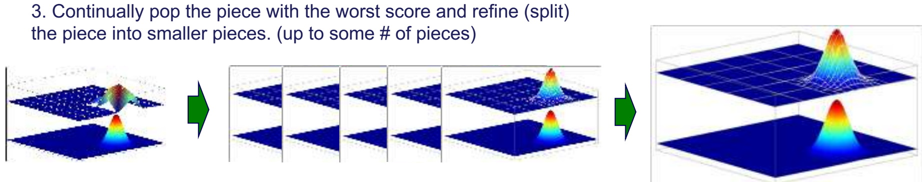
1. Make an initial uniform function



2. Maintain a fixed-size priority queue of pieces with poor regression scores



3. Continually pop the piece with the worst score and refine (split) the piece into smaller pieces. (up to some # of pieces)



The Reflector Tool

Smart Refinement

Typical code structure using the Smart Refinement option.

```
01 OF_Reflector reflector(aof);
02
03 reflector.setParam("uniform_amount", 1000);
04 reflector.setParam("smart_amount", 400);
05 reflector.setParam("refine_thresh", 0.5);
06 reflector.create();
07
08 IvPFunction *ipf = reflector.extractIvPFunction();
```

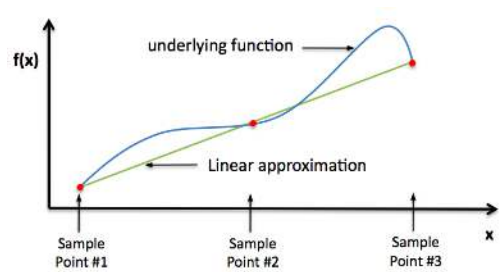
- 01 Create a reflector passing it the underlying function
- 03 Specify the initial amount of uniform pieces
- 04 Specify the additionaly amount of pieces dedicated to smart refinement
- 05 Optionally specify a regression threshold to abort smart refinement early
- 06 Invoke the algorithm
- 08 Extract the objective function

The Reflector Tool

Pros and Cons of Smart Refinement

- The appeal of smart refinement is that it is efficient in its use of pieces and sampling of the underlying function.
- Less pieces, greater accuracy over pure uniform functions.
- Does not require the user to know anything about the underlying function.

- The drawback is that its regression scores are not always accurate.



- In short, Smart Refinement is very powerful and convenient if used in conjunction with other refinement, and not relied on too heavily.

MITMECHE

THE
END