Multi-Objective Optimization for Virtual Machine Allocation and Replica Placement in Virtualized Hadoop

CARLOS GUERRERO , ISAAC LERA , BELEN BERMEJO , AND CARLOS JUIZ , SENIOR MEMBER, IEEE

PRESENTED BY: PAYMAN KHANI

Overview

Introduction

- The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. (HDFS: splitting and storing files, MapReduce: Job schedule)
- The policies for *selecting the features of the VMs* and *distributing them in PMs* and *the chunk replicas in DataNodes* are commonly known as :
  - VM template selection,
  - VM allocation,
  - replica placement.
- Important questions arise in this distribution process:
  - How many VMs are necessary to deploy the HDFS system?
  - Which are the best features for the VMs?
  - Which PMs should allocate the VMs?
  - Where should the chunk replicas be stored?
- The management policies answer all these questions. An efficient implementation of these policies has a direct impact on the system performance and on the resource usages.
- However, the optimal solution cannot be directly calculated because it is an NP-hard problem and all the possible placement combinations should be measured.

Related Work

The related works are organized in two parts:

- Evolutionary approaches for VM management
- Replica placement works in HDFS

- Kessaci et al. presented a **Pareto multi-objective** version of the energy-aware multi-start local search algorithm dealing with energy consumption and SLAs. The algorithm allocates the VMs by reducing the response time of the jobs inside the machines and the energy consumption in the physical level.
  Available: http://dblp.unitrier.de/db/conf/ieeehpcs/ieeehpcs2012.html#KessaciMT12

- Adamuthe et al. compared genetic algorithms with non-dominated sorting genetic algorithms (NSGA) to maximize physical resource usages, the balanced distribution of VMs among physical machines and the wasted resources. Their work is probably the **most similar work to this approach in terms of VM allocation**.
Available: http://dx.doi.org/10.1109/CUBE.2013.12p

- There are several efforts addressing HDFS placement strategies. **The most similar work to our approach in terms of replica placement** is probably the work of Long et al.
Available: http://www.sciencedirect.com/science/article/pii/S1383762113002671

Problem Statement and Formulation

- **System Modeling**
- **Optimization objectives of the proposal**

Problem Statement and Formulation

System Modeling:

The system is defined by:

(i)

The characteristics of:

- Physical machines
- Virtual machines
- HDFS file system;

(ii)

- Allocation relationships between VMs and PMs, and
- Placement relationship between the replicas and VMs.

Problem Statement and Formulation

System Modeling (i) :

➢ *Physical machines:* Each PM $pm_i \in PM$ is characterized by :

- *Capacity of its resources*: Resource model is limited to three components - CPU, disk bandwidth and network bandwidth.

$$pmResCap_{pm_i} = \langle pmResCap^{cpu}_{pm_i}, pmResCap^{diskbw}_{pm_i}, pmResCap^{netbw}_{pm_i} \rangle$$

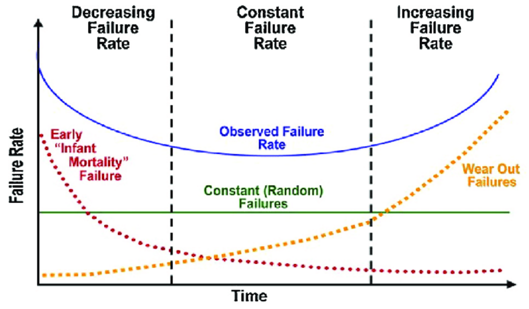
- *Power feature* :

$$pmPowFeat_{pm_i} = \langle pmPowMin_{pm_i}, pmPowMax_{pm_i}, \alpha_{pm_i}, \beta_{pm_i}, \delta_{pm_i}, \gamma_{pm_i} \rangle$$

Where the first two are the minimum and maximum power consumptions of the PM, respectively, and the others are model coefficients.

- *Failure metrics*:

$$pmFail_{pm_i} = \langle pmFail^{min}_{pm_i}, pmFail^{max}_{pm_i} \rangle$$



Failures are modeled as a bathtub curve with respect to the CPU usage of the machine.
https://en.wikipedia.org/wiki/Bathtub_curve , https://ieeexplore.ieee.org/abstract/document/6903562

Problem Statement and Formulation

System Modeling (i):

➢*Virtual machines*: Here defines DataNodes as VM instances.

$vm_n \in VM$ are characterized by their VM instance type, $vmt_t \in VMType$,

In particular, it defines:

- *Resource capacities:*

$$vmResCap_{vm_n} = \langle vmResCap^{cpu}_{vm_n}, vmResCap^{diskbw}_{vm_n}, vmResCap^{netbw}_{vm_n} \rangle$$

- *Failure metrics:*

$$vmFail_{vm_n} = \langle vmFail^{min}_{vm_n}, vmFail^{max}_{vm_n} \rangle$$

In this paper VM refers to VM instance, and template refers to VM instance type.

Problem Statement and Formulation

System Modeling (i):

➢*HDFS (Hadoop distributed file system)*:
HDFS is a distributed and scalable file system in which files are split into blocks (chunks) that are stored across the nodes of a cluster (DataNodes) or VMs here.

- A file $f_u \in HDFS$ is defined as concatenation of its chunks/blocks:

$$f_u = fb_0^u \parallel fb_1^u \parallel fb_2^u \parallel .. \parallel fb_{fC_{f_u}-1}^u$$

Where the number of chunks, is determined by the file size and the chunk size as:

$$fC_{f_u} = \lceil \frac{fSize_{f_u}}{fbSize_{f_u}} \rceil$$

- To guarantee the availability of the files, HDFS replicates each chunk across several DataNodes (VMs). We define a file chunk as a set of replicas:

$$fb_x^u = \{fb_x^u(0), fb_x^u(1), \ldots, fb_x^u(fR_{f_u} - 1)\}$$

where fR_{f_u}, the replication factor, is also set individually for each file.

Problem Statement and Formulation

System Modeling (i):

➢*HDFS (Hadoop distributed file system)*:

- Replicas are characterized by the resource consumption that the execution of MapReduce jobs (MR) generates.
- Apache Hadoop distributes the jobs across the DataNodes to avoid moving the data.
  Replica resource consumption:

$$repResCon_{fb_x^u[r]} = \langle repResCon_{fb_x^u[r]}^{cpu}, repResCon_{fb_x^u[r]}^{diskbw}, repResCon_{fb_x^u[r]}^{netbw} \rangle$$

- $repAccRate_{fb_x^u[r]}$ (MR job's access rate for a replica)

  The total workload in a DataNode (VMs) depends on the *access rate* of each replica.

Problem Statement and Formulation

System Modeling (ii):

- Allocation:

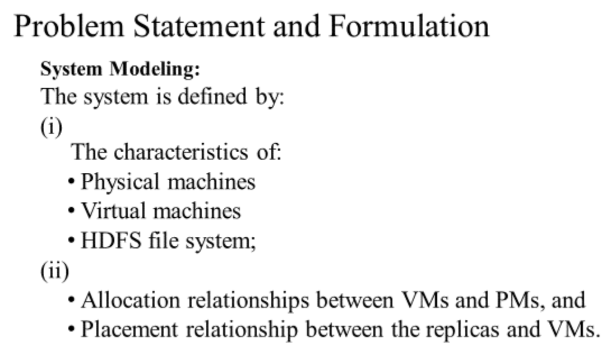
The VMs are deployed in PMs.

$$allocation \ : \ VM \rightarrow PM.$$

- Placement:

The storage of the replicas in the DataNodes (VMs).

$$placement \ : \ \{fb_x^u[r]\} \rightarrow VM$$



Problem Statement and Formulation

System Modeling:

The system is defined by:

(i)

The characteristics of:

- Physical machines
- Virtual machines
- HDFS file system;

(ii)

- Allocation relationships between VMs and PMs, and
- Placement relationship between the replicas and VMs.

Problem Statement and Formulation

System Modeling:

- *The total resource consumption of a VM* depends on the access rates of the replicas it places and on the consumption generated in those accesses. Consequently, the **VM resource consumption** can be calculated as:

$$vmResCon_{vm_n} = \sum_{fb_x^u[r]} (repResCon_{fb_x^u[r]} \times repAccRate_{fb_x^u[r]}),$$
$$\forall\ fb_x^u[r] \mid placement(fb_x^u[r]) = vm_n.$$

- *The **PMs' resource consumption*** can be calculated considering the VMs' resource consumptions and the allocation relationships:

$$pmResCon_{pm_i} = \sum_{vm_n} vmResCon_{vm_n},$$
$$\forall\ vm_n \mid allocation(vm_n) = pm_i.$$

Problem Statement and Formulation

System Modeling:

- Additionally *Hypervisor, or VM monitor (VMM),* installed in the PM also consumes computational resources. This overhead is represented in this model by: $hypResCon_{pm_i}$

It assumed that this overhead is constant.

- In summary, the **PM resource consumption is generated by the MapReduce jobs executed in:**
- The allocated VMs and
- The overhead of the VMM.

$$pmResCon_{pm_i} + hypResCon_{pm_i}$$

Problem Statement and Formulation

System Modeling:

- *Utilization of the physical machines* (pmU_{pm_i}):

The utilization is a metric that measures the ratio between the consumption and the available capacity of a system resource, and its value, which ranges between 0.0 and 1.0, is calculated as:

$$pmU_{pm_i} = \frac{pmResCon_{pm_i}}{pmResCap_{pm_i}}$$

- *Three constraints:*

The first one is related to the built-in placement policy of HDFS that does not allow DataNodes (VMs) to store the same replica twice.

$$placement(fb_x^u[r]) \neq placement(fb_x^u[r']) \quad \forall\ fb_x^u[r], fb_x^u[r'] \in fb_x^u.$$

The second and third constraints limit the total consumption of resources in a PM, or in a VM, to be smaller than the available capacities:

$$pmResCon_{pm_i} + hypResCon_{pm_i} < pmResCap_{pm_i}, \quad \forall\ pm_i \in PM$$

$$vmResCon_{vm_n} < vmResCap_{vm_n}, \forall\ vm_n \in VM.$$

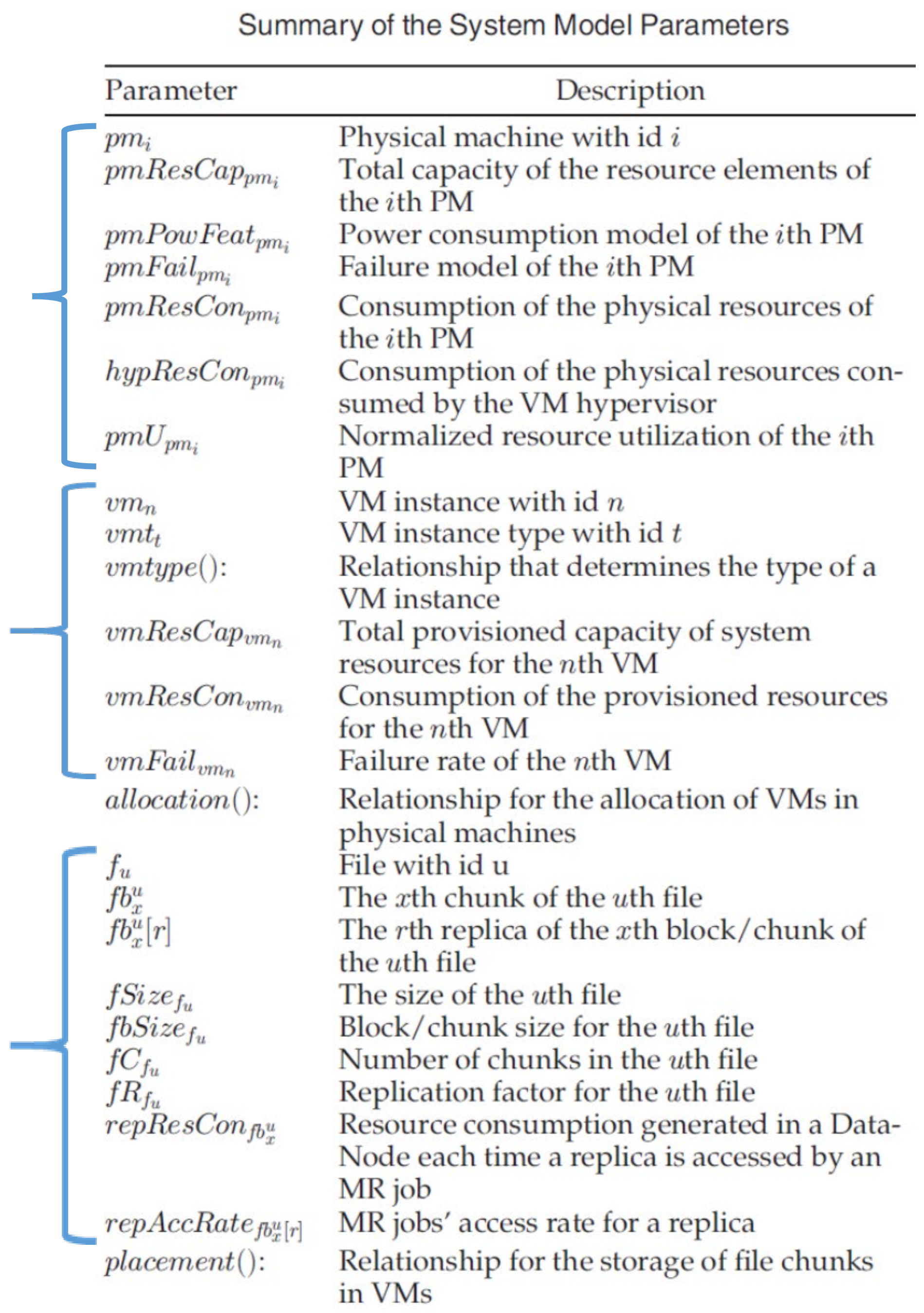
Problem Statement and Formulation

System Modeling:

PM related

VM related

File related



Summary of the System Model Parameters

Parameter	Description
pm_i	Physical machine with id i
$pmResCap_{pm_i}$	Total capacity of the resource elements of the ith PM
$pmPowFeat_{pm_i}$	Power consumption model of the ith PM
$pmFail_{pm_i}$	Failure model of the ith PM
$pmResCon_{pm_i}$	Consumption of the physical resources of the ith PM
$hypResCon_{pm_i}$	Consumption of the physical resources consumed by the VM hypervisor
pmU_{pm_i}	Normalized resource utilization of the ith PM
vm_n	VM instance with id n
vmt_t	VM instance type with id t
$vmtype()$:	Relationship that determines the type of a VM instance
$vmResCap_{vm_n}$	Total provisioned capacity of system resources for the nth VM
$vmResCon_{vm_n}$	Consumption of the provisioned resources for the nth VM
$vmFail_{vm_n}$	Failure rate of the nth VM
$allocation()$:	Relationship for the allocation of VMs in physical machines
f_u	File with id u
fb^u_x	The xth chunk of the uth file
$fb^u_x[r]$	The rth replica of the xth block/chunk of the uth file
$fSize_{f_u}$	The size of the uth file
$fbSize_{f_u}$	Block/chunk size for the uth file
fC_{f_u}	Number of chunks in the uth file
fR_{f_u}	Replication factor for the uth file
$repResCon_{fb^u_x}$	Resource consumption generated in a DataNode each time a replica is accessed by an MR job
$repAccRate_{fb^u_x[r]}$	MR jobs' access rate for a replica
$placement()$:	Relationship for the storage of file chunks in VMs

Problem Statement and Formulation

System Modeling:

➢The performance metrics of a virtualized Hadoop are strongly influenced by:

- The allocation of the VMs, allocation()
- The selection of the VM types, vmType()
- The allocation of the replicas, placement()
- The number of VMs, |VM|
- The chunk sizes, $fbSize_{fu}$.
- The replication factor (how many copy), fR_{fu}

➢These parameters are <u>customized by the system administrator to optimize</u>, for example, resource usage, power consumption, or data availability.

➢Now lets talk about the optimization objectives of the proposal.

Objective:

- Resource Waste
- Power Consumption
- Data Unavailability

Problem Statement and Formulation

Optimization objectives of the proposal

Problem Statement and Formulation

➢**Resource Waste Objective:**

- The proposed strategy is not only to <u>allocate as many VMs as possible</u> but also to <u>balance the consumption of the resources</u>.
- Example of resource waste :

A PM that has provisioned 95% of its main memory and 30% percent of its CPU

- The resource waste is 0 when no VM is allocated since the PM could be switched off.
- $\sigma(pmU_{pm_i})$ is the standard deviation of the three utilizations for CPU, disk bandwidth and network bandwidth. It represents how balanced the consumption of the resources is.
- $\sum pmU_{pm_i}$ is the sum of those three values, and it represents the usage of the resources.
- ε adjusts the weight of resource usage and resource balancing in the calculation. The smaller the value of the parameter is, the greater the importance that is given to the resource balance in front of the usage.
  They considered ε = 0.15 after the evaluation of several values.

$$ResourceWaste(pm_i) = \begin{cases} 0, & if\ \nexists\ vm_n \mid allocation(vm_n) = pm_i \\ \frac{\sigma(pmU_i)+\varepsilon}{\sum pmU_{pm_i}}, & otherwise \end{cases}$$

$$\sigma(pmU_{pm_i}) = \sigma(pmU_{pm_i}^{cpu}; pmU_{pm_i}^{diskbw}; pmU_{pm_i}^{net})$$

$$\sum pmU_{pm_i} = pmU_{pm_i}^{cpu} + pmU_{pm_i}^{diskbw} + pmU_{pm_i}^{net}.$$

Problem Statement and Formulation

➢**Power Consumption Objective:**

PM power consumption by considering the CPU, the network bandwidth and the storage bandwidth is modeled :

$$PowCons(pm_i) = \boxed{PowCons^{cpu}(pm_i)} + PowCons^{net}(pm_i) + PowCons^{disk}(pm_i).$$

- *CPU power consumption* model was defined as a piecewise linear relationship with the CPU load.

$$PowCons^{cpu}(pm_i) = \begin{cases} PowCons^{cpu-low}(pm_i), & if\ pmU^{disk}_{pm_i} \leq \mathcal{L}_{pm_i} \\ PowCons^{cpu-high}(pm_i), & if\ pmU^{disk}_{pm_i} \geq \mathcal{L}_{pm_i} \end{cases}$$

$$PowCons^{cpu-low}(pm_i) = \alpha_{pm_i} \times (pmPowMax_{pm_i} - pmPowMin_{pm_i}) \times pmU^{cpu}_{pm_i}$$

$$PowCons^{cpu-high}(pm_i) = \beta_{pm_i} \times (pmPowMax_{pm_i} - pmPowMin_{pm_i}) + (1-\beta_{pm_i}) \times (pmPowMax_{pm_i} - pmPowMin_{pm_i}) \times pmU^{cpu}_{pm_i},$$

where $\mathcal{L}_{pm_i}$ is the load at which the power consumption trend changes on pm_i and α_{pm_i} and β_{pm_i} are the coefficients for low and high CPU load levels, respectively.

Problem Statement and Formulation

➢**Power Consumption Objective:**

$$PowCons(pm_i) = PowCons^{cpu}(pm_i)$$
$$+\boxed{PowCons^{net}(pm_i) + PowCons^{disk}(pm_i).}$$

$$PowCons^{cpu}(pm_i) = \begin{cases} PowCons^{cpu-low}(pm_i), & if\ pmU^{disk}_{pm_i} \leq \mathcal{L}_{pm_i} \\ PowCons^{cpu-high}(pm_i), & if\ pmU^{disk}_{pm_i} \geq \mathcal{L}_{pm_i} \end{cases}$$

$$PowCons^{cpu-low}(pm_i) = \alpha_{pm_i} \times (pmPowMax_{pm_i} - pmPowMin_{pm_i}) \times pmU^{cpu}_{pm_i}$$

$$PowCons^{cpu-high}(pm_i) = \beta_{pm_i} \times (pmPowMax_{pm_i} - pmPowMin_{pm_i}) + (1-\beta_{pm_i}) \times (pmPowMax_{pm_i} - pmPowMin_{pm_i}) \times pmU^{cpu}_{pm_i},$$

- <u>*Network and storage power*</u> models were defined as linear relationships with the bandwidth usage.

$$PowCons^{disk}(pm_i) = \delta_{pm_i} \times (pmPowMax_{pm_i} - pmPowMin_{pm_i}) \times pmU^{disk}_{pm_i}$$

$$PowCons^{net}(pm_i) = \gamma_{pm_i} \times (pmPowMax_{pm_i} - pmPowMin_{pm_i}) \times pmU^{net}_{pm_i},$$

where δ_{pm_i} and γ_{pm_i} are the model coefficients.

Problem Statement and Formulation

➢**Data Unavailability Objective:**

- Considering a virtualized Hadoop, a DataNode fails when either the PM or the VM fails. Thus, a replica becomes unavailable with a failure in:
  - The VM placing it or
  - The PM allocating this VM
- Data unavailability is reduced whether the VMs with the replicas of the same chunk are allocated in different PMs.

$$Unavailability(f_u) = \sum_{b_x^u \in b_u} BlockUnavailability(b_x^u)$$

The unavailability of a chunk, FailureRate , is represented as

$$BlockUnavailability(b_x^u) = \prod_{pm_i \in PM(b_x^u)} \left(pmFail_{pm_i} + \prod_{vm_n \in A} vmFail_{vm_n} \right)$$

Problem Statement and Formulation

➢**Data Unavailability Objective:**

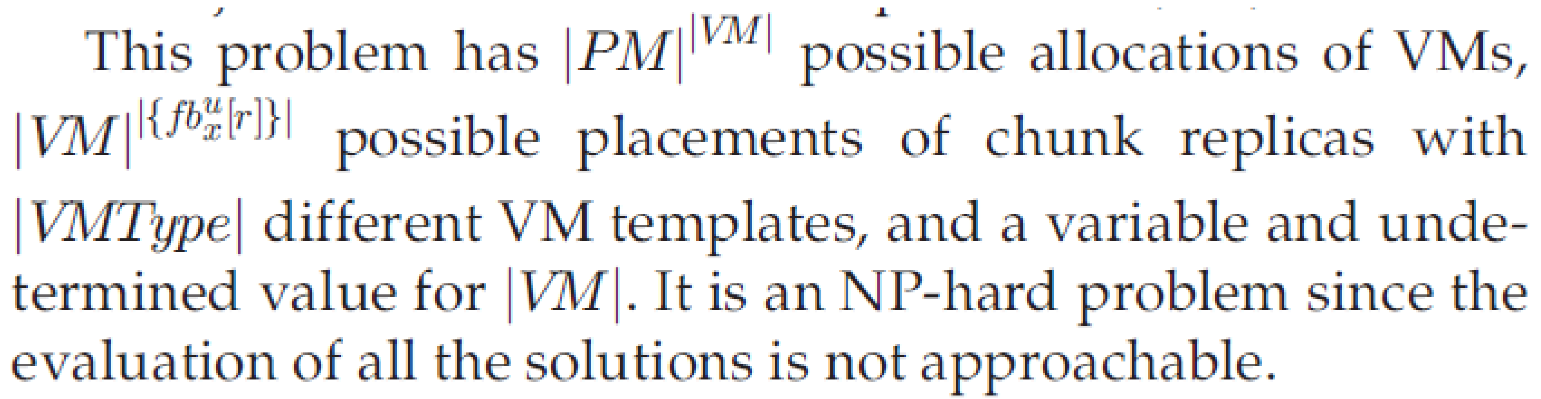
$$BlockUnavailability(b_x^u) = \prod_{pm_i \in PM(b_x^u)} \left(pmFail_{pm_i} + \prod_{vm_n \in A} vmFail_{vm_n} \right)$$

- The outer multiplication reflects that a chunk is unavailable when all the PMs that allocate VMs storing the replicas of the chunk (b_x^u) are unavailable. $PM(b_x^u)$ is the set of PMs with at least one VM storing the chunk.
- The first term of the formula inside the parenthesis, represents the cases when the chunk replicas in the PM become unavailable due to a failure in that PM.
- The second term, the inner multiplication, represents the cases when the chunk replicas in the PM become unavailable due to failures in all the VMs (A) that contain them.

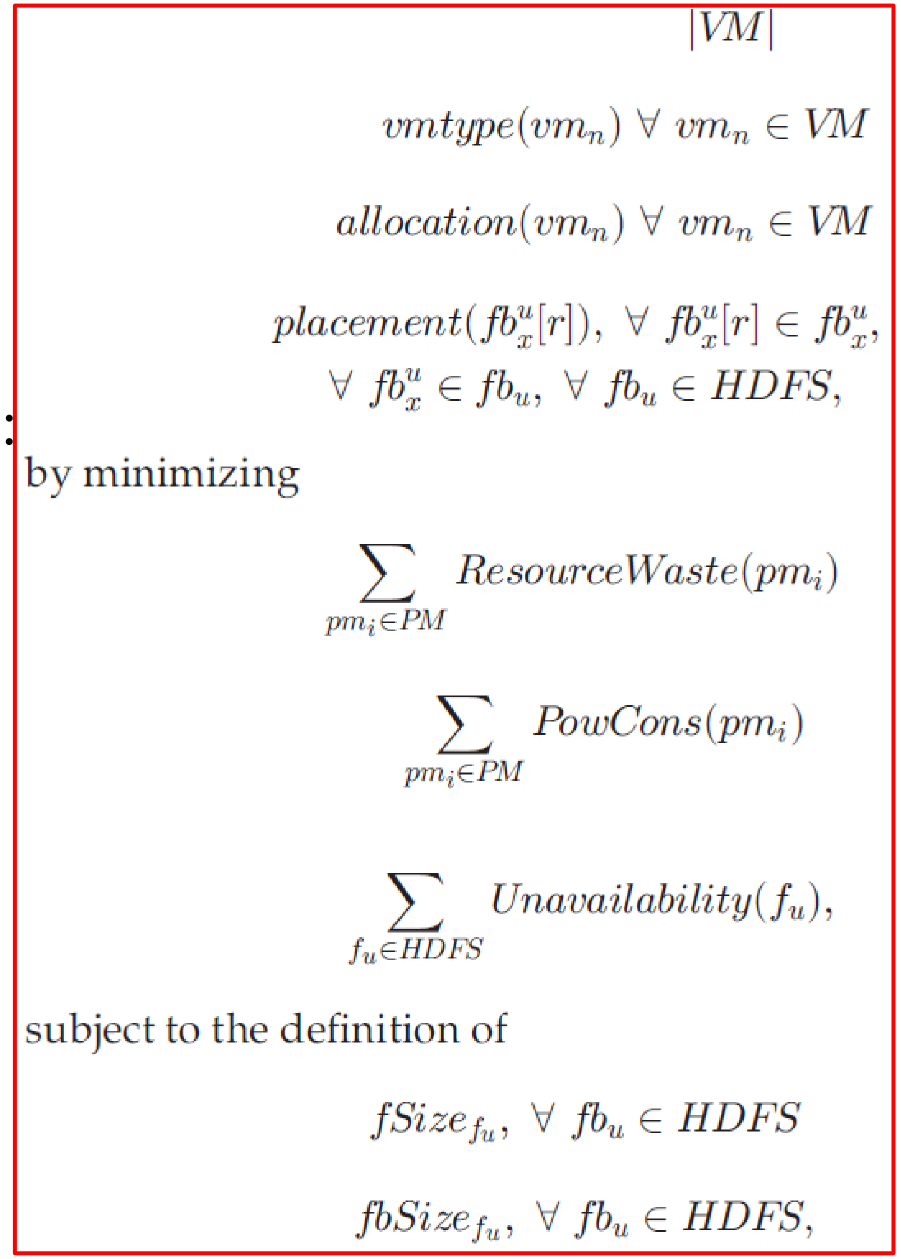
Problem Statement and Formulation

➢**Optimization Formulation:**

- The objective of the optimization is to minimize:
  1) File unavailability 2) Power consumption
  3) Waste of resources
- The decision variables, i.e., the elements to be managed, are:
  - Allocation of the VMs,
  - Selection of the VM templates
  - Number of VM instances
  - Placement of the replicas
- The problem is formally defined here :



This problem has $|PM|^{|VM|}$ possible allocations of VMs, $|VM|^{|\{fb^u_x[r]\}|}$ possible placements of chunk replicas with $|VMType|$ different VM templates, and a variable and undetermined value for $|VM|$. It is an NP-hard problem since the evaluation of all the solutions is not approachable.



$|VM|$

$vmtype(vm_n)\ \forall\ vm_n \in VM$

$allocation(vm_n)\ \forall\ vm_n \in VM$

$placement(fb^u_x[r]),\ \forall\ fb^u_x[r] \in fb^u_x,$

$\forall\ fb^u_x \in fb_u,\ \forall\ fb_u \in HDFS,$

by minimizing

$$\sum_{pm_i \in PM} ResourceWaste(pm_i)$$

$$\sum_{pm_i \in PM} PowCons(pm_i)$$

$$\sum_{f_u \in HDFS} Unavailability(f_u),$$

subject to the definition of

$fSize_{f_u},\ \forall\ fb_u \in HDFS$

$fbSize_{f_u},\ \forall\ fb_u \in HDFS,$

Genetic Algorithm Proposal

Genetic Algorithm Proposal

- They proposed using the *non-dominated sorting genetic algorithm-II (NSGA-II)*to solve this multi-objective optimization problem.
- Genetic algorithms (GAs) are metaheuristic approaches for solving NP-hard optimizations, and NSGA-II is one of the most common algorithms when the optimization has multiple objectives to minimize.
- A solution is called non-dominated, Pareto optimal, Pareto efficient or non-inferior, if none of the objective functions can be improved in value without degrading some of the other objective values.
- Multi-objective optimization also known as multi-objective programming, vector optimization, multi-criteria optimization, multi-attribute optimization or Pareto optimization.
- A solution is called Pareto optimal, if there does not exist another solution that dominates it. The set of Pareto optimal outcomes is often called the Pareto front, Pareto frontier, or Pareto boundary.

Genetic Algorithm Proposal

➢More resources:

- Multi-Objective Optimization using Non-Dominated Sorting Genetic Algorithm with Numerical Example Step-by-Step:

https://www.slideshare.net/AhmedGadFCIT/multiobjective-optimization-using-nondominated-sorting-genetic-algorithm-with-numerical-example-stepbystep

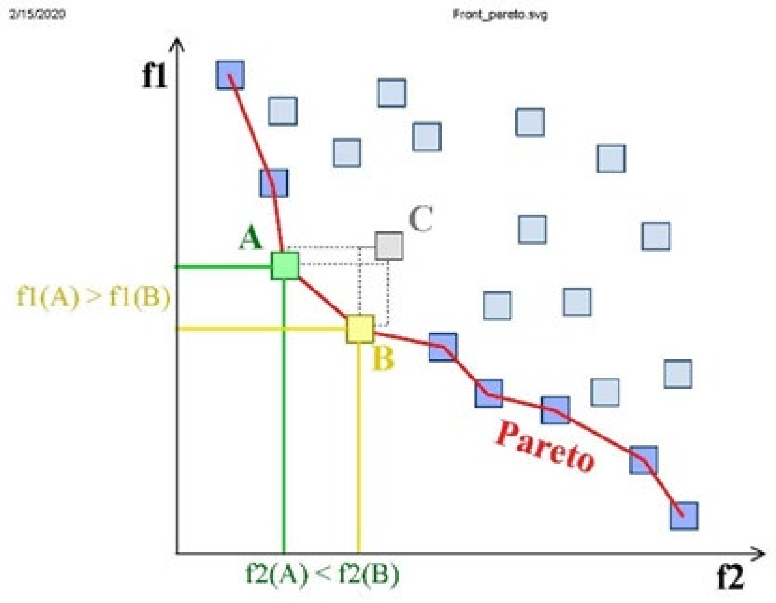
- Video resources for NSGA-II and GA.

https://www.youtube.com/watch?v=SL-u_7hIqjA

https://www.youtube.com/watch?v=JgqBM7JG9ew

- Video resource for Pareto Optimality.

https://www.youtube.com/watch?v=cT3DcuZnsGs



Genetic Algorithm Proposal

- The implementation of a GA involves defining the:
  - **Chromosomes,**
  - Fitness function,
  - **Crossover and mutation operators,**
  - Selection operator, the replacement operator (offspring generation),

- NSGA-II sets the selection and replacement operators, and it defines the fitness function as a vector that includes all the objective functions.
- The following sections explain the implementation details that are not preset by NSGA-II.

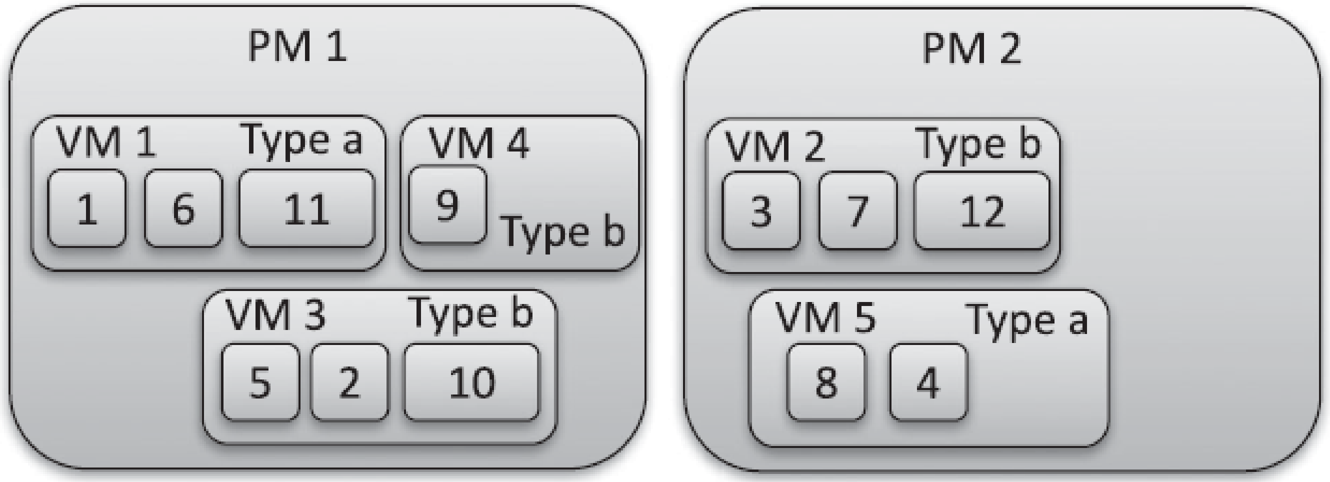
Genetic Algorithm Proposal

Chromosome Representation:

- In our case, the individuals (solutions) of the population represent
  - VM allocation, VM type selection
  - Replica placement,
- Considering a fixed number of replicas and a variable number of VMs. We represent each individual with two arrays:
  - vm-chromosome (Cvm) : for allocation() and vmtype() relationships
  - block-chromosome (Cblock) : for placement() relationship.
- The length of the vm-chromosome can change between solutions since there will be solutions with different numbers of VMs.
- Block-chromosome has a fixed length since the chunk size $(fbSize_{f_u})$ and the replication factor (fR_{f_u}) are equal and constant for all the files, and it is known before the optimization process.

Genetic Algorithm Proposal

Chromosome Representation:



vm-chromosome

	1	2	3	4	5
allocation	1	2	1	1	2
template type	a	b	b	b	a

block-chromosome

1	2	3	4	5	6	7	8	9	10	11	12
1	3	2	5	3	1	2	5	4	3	1	2

Genetic Algorithm Proposal

Crossover and Mutation Operators:

- The *crossover operator* defines how two parent solutions are combined to obtain two evolved children.
- Since we are dealing with three decision variables (vm allocation, vm template selection, and replica placement), the crossover is sequentially divided into:
  - ✓ *First phase :* where the crossover of the vm-chromosome is performed over the allocation and template-type arrays
  - ✓ *Second phase :* for the crossover of the block-chromosome.
- Use one cutting-point crossover operator in both cases.

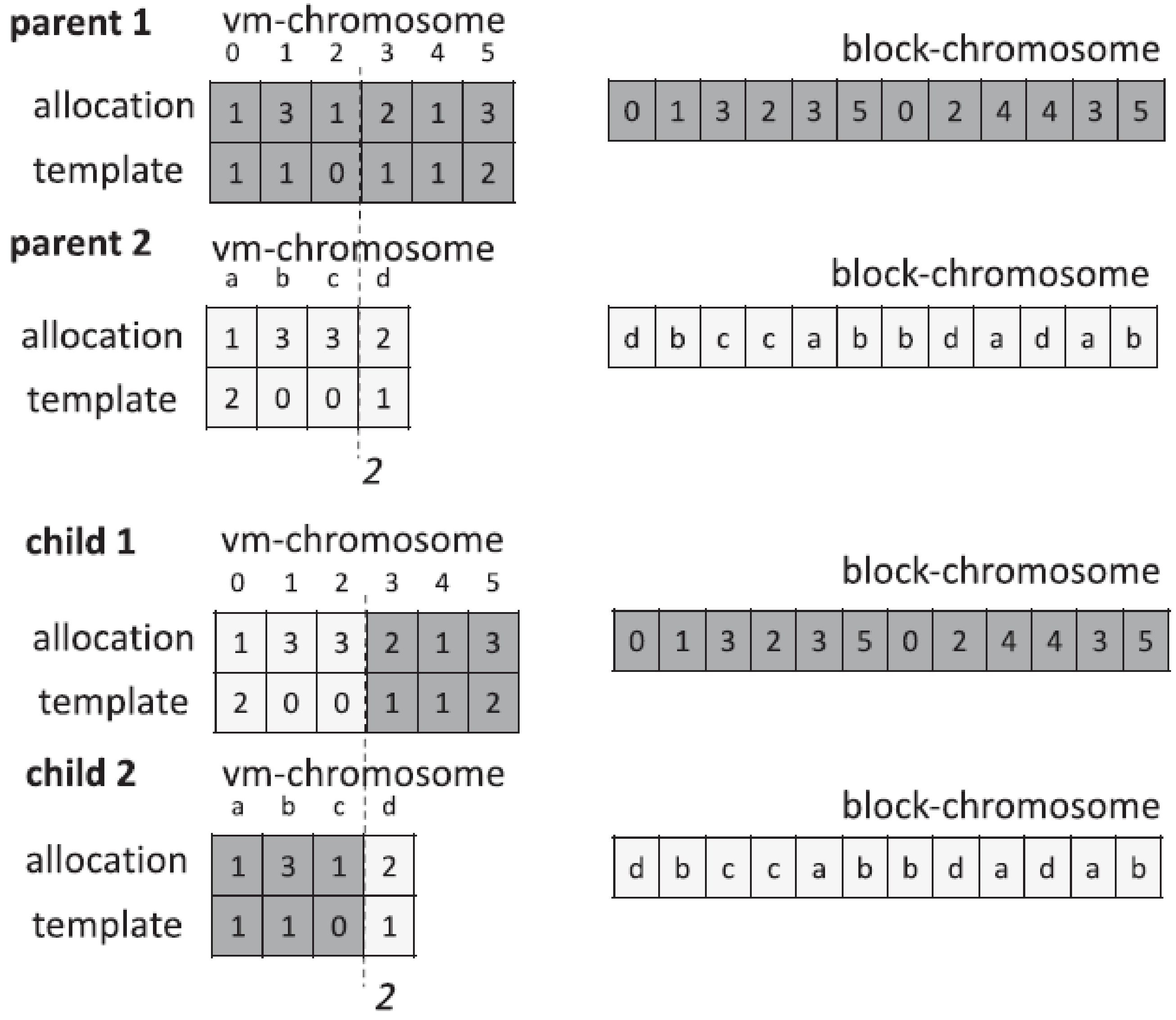
This operator generates one random cutting point that splits the chromosomes into two pieces. Children's chromosomes are generated by combining the opposite pieces from both parents.

Genetic Algorithm Proposal

Crossover and Mutation Operators:

First Phase of the crossover:

- The one-point operator for variable chromosome length is applied to the vm-allocation:
- One random number is generated between 0 and the minimum length of both parents' chromosomes, and it splits the allocation and template arrays into two pieces.
- The first child is generated by combining the left piece of one parent with the right piece of the other.
- Similarly, the second child is generated with the remaining pieces.

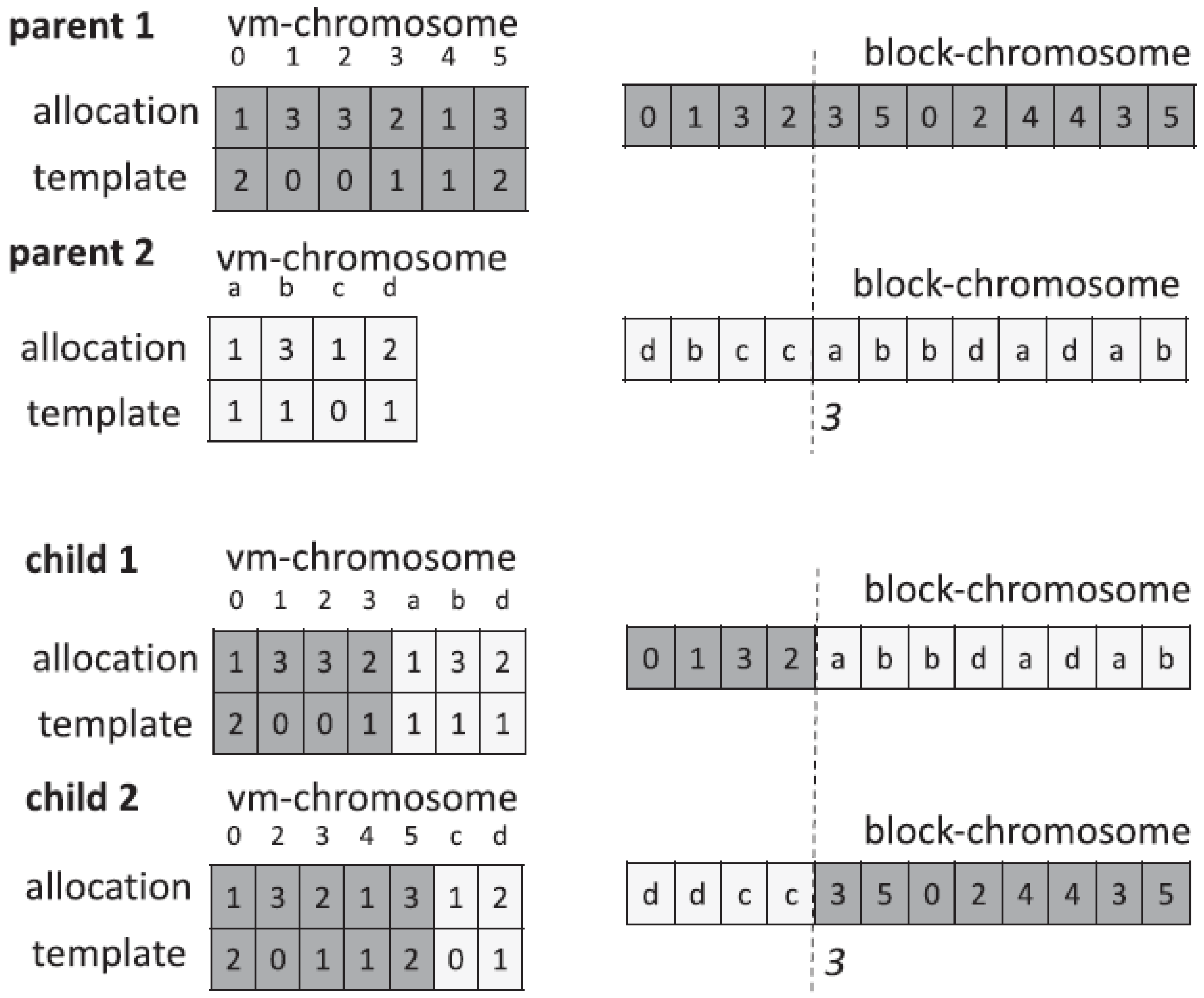


Genetic Algorithm Proposal

Crossover and Mutation Operators:

Second phase of the crossover:

- The second phase of the crossover is more complex since the combination of the block-chromosomes results in solutions with replicas allocated in VMs from both fathers.
- Consequently, all the VMs referenced from the new block-chromosome should be incorporated in the resulting vm-chromosome.
- In this example, the new block-chromosome of the first child results in placing all the replicas in VMs 0, 1, 2, and 3 from the first parent (dark gray positions in the vm-chromosme) and VMs a, b, and d from the second (light gray positions)



Genetic Algorithm Proposal

Crossover and Mutation Operators:

Mutation Operators:

- Mutation operators generate random changes in the chromosomes of a new solution in the offspring. We define mutations for both vm and block chromosomes.
  - ✓ First, in the VM mutation phase:
    VM growth and shrink mutations respectively increase or decrease the number of VMs with a probability of 1/3 .
    Then, the VM replace mutation iterates the positions of both vm-chromosome arrays (allocation and template) and randomly changes their values with a probability of 0.5.
    On average, half of the VM allocations and template types will be modified in the mutated solution.

  - ✓ Block replace mutation is applied to the block-chromosome by also iterating the positions of the chromosome and randomly changing their values with a probability of 0.5.
    The new random placements of the chunks are generated by only considering the current VMs in the solution.
    Therefore, the block replace mutation does not generate changes in the length of the vm-chromosome.

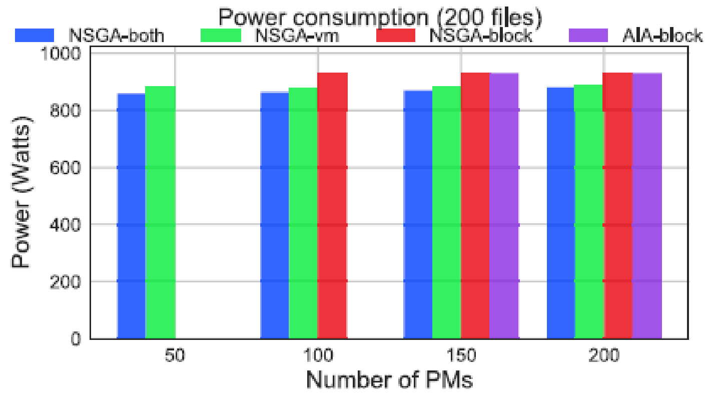
Genetic Algorithm Proposal

Genetic Algorithm Setting:

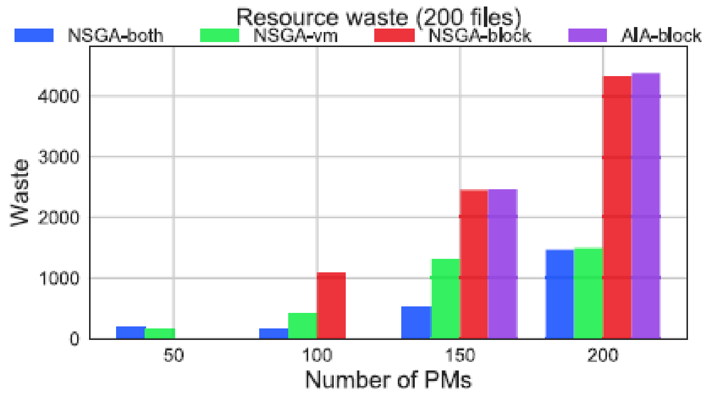
- The execution of a GA needs to set up some parameters that cannot be generalized between experimental domains, such as the ending condition, the generation of the initial population and so forth.
- These parameters are commonly set by exploration of cases in a previous execution phase.
- In addition, the initial replica placement is performed in a round-robin distribution. The allocation of the VMs is also performed in a round-robin distribution across the PMs. two replicas cannot be placed in the same DataNode.

Experiment design, Result and discussions:

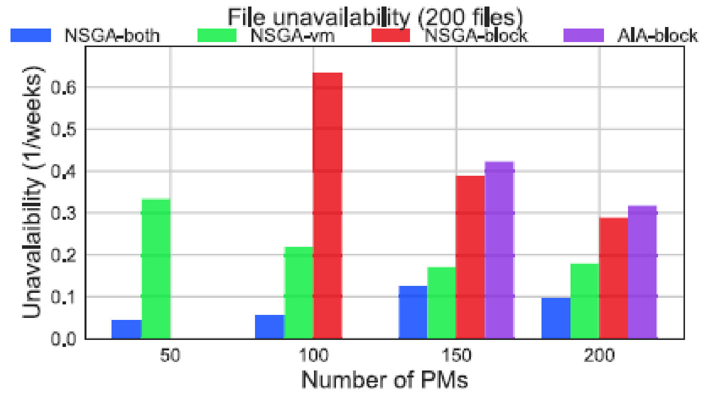
- Through the iterative execution of NSGA-II, the Pareto optimal front evolved until the finish condition was reached.
- *The Pareto optimal front* is the set of solutions that minimize our objective functions. The program calculated the values of the objective functions for each individual.
- *The output of a multi-objective optimization is a Pareto optimal front*. This is a set that includes the solutions that are not dominated by any other.
- A non-dominated solution has at least one objective with a smaller value than all the other solutions.



(a) Optimized values for power consumption.

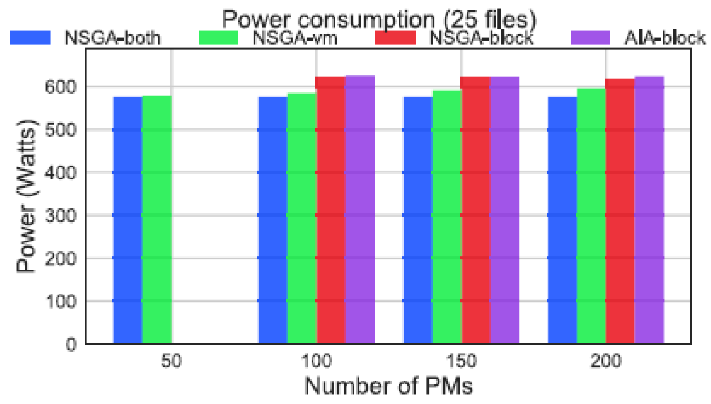


(b) Optimized values for resource waste.

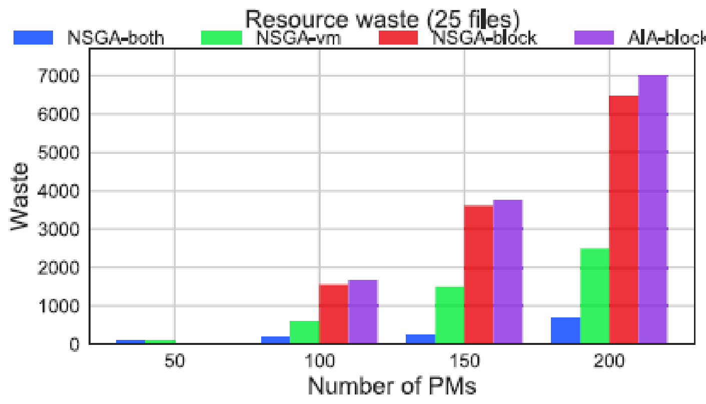


(c) Optimized values for file unavailability.

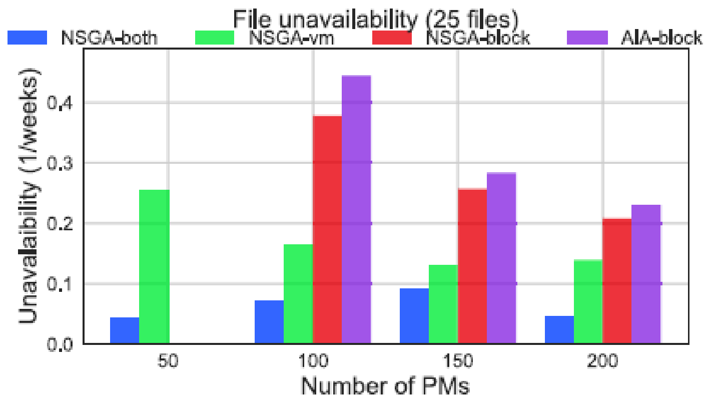
Experiment design, Result and discussions:



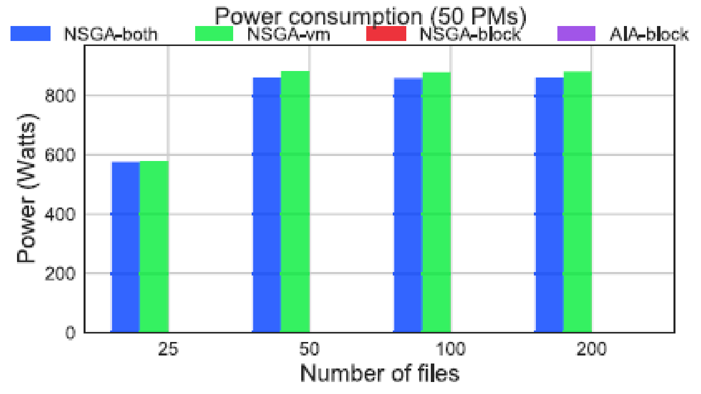
(a) Optimized values for power consumption.



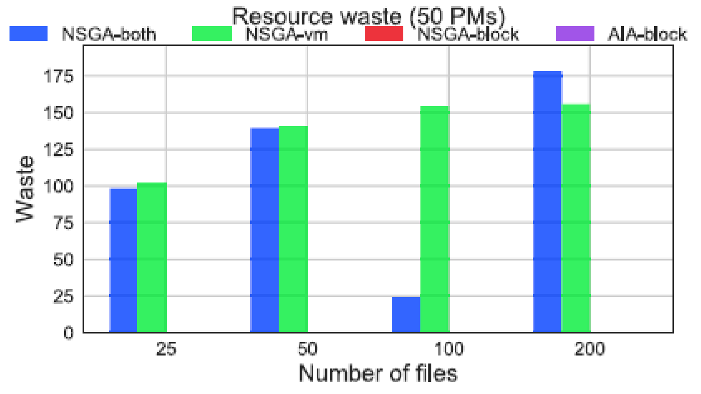
(b) Optimized values for resource waste.



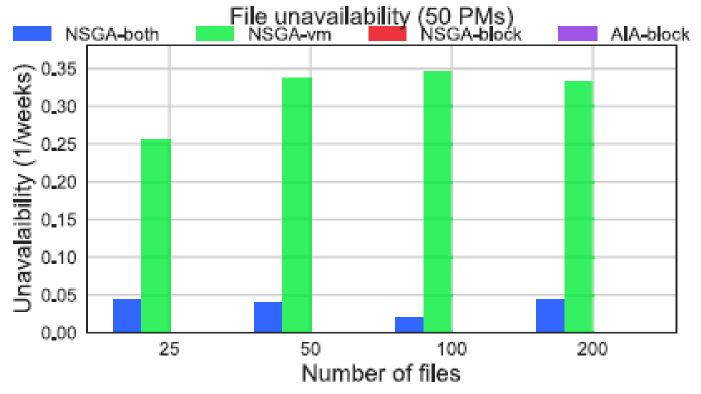
(c) Optimized values for file unavailability.



(a) Optimized values for power consumption.

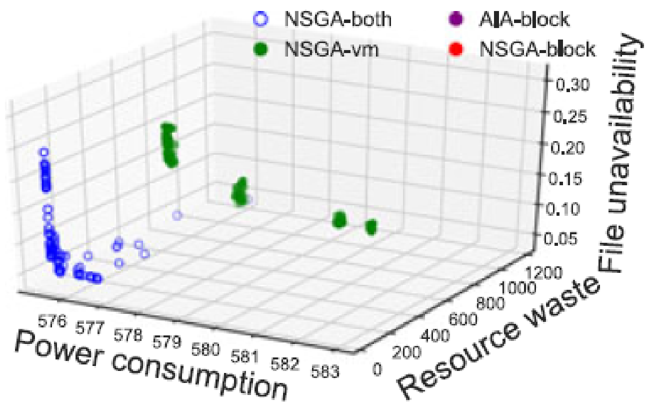


(b) Optimized values for resource waste.

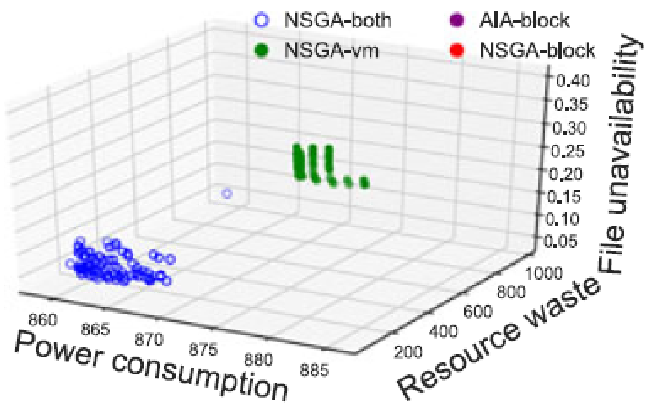


(c) Optimized values for file unavailability.

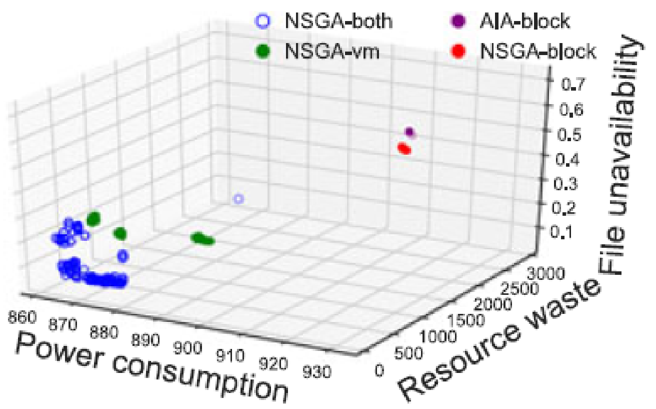
Experiment design, Result and discussions:



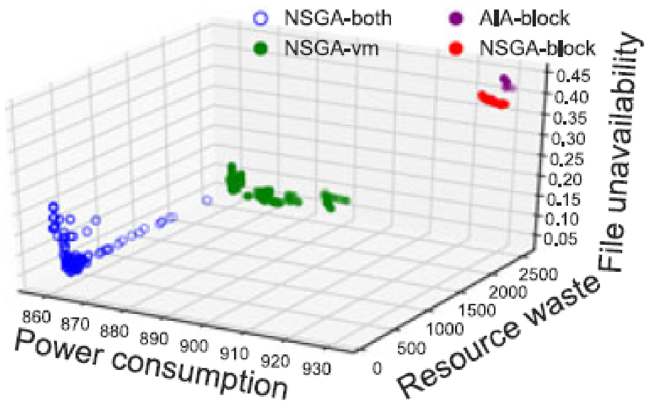
(a) Experiment size of 50 PMs and 25 files.



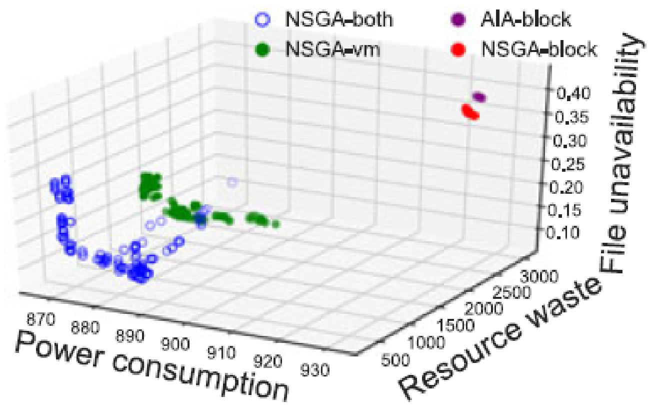
(b) Experiment size of 50 PMs and 200 files.



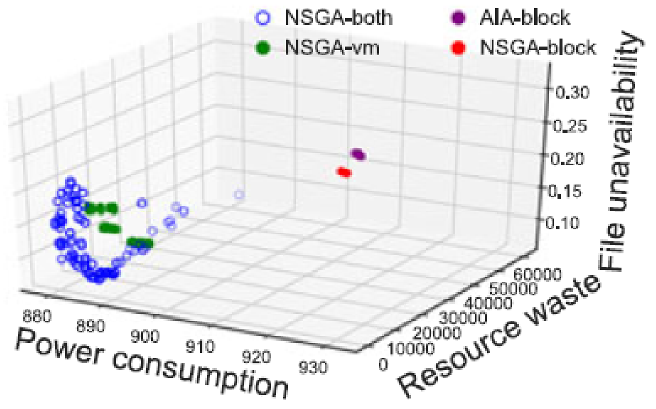
(c) Experiment size of 100 PMs and 200 files.



(d) Experiment size of 150 PMs and 50 files.

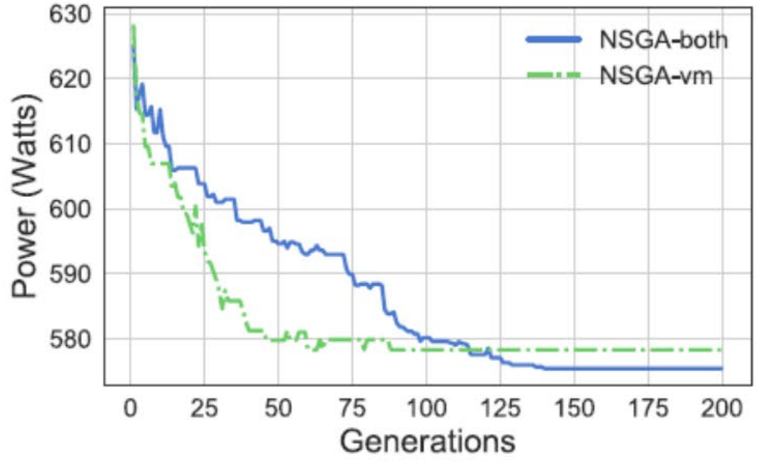


(e) Experiment size of 150 PMs and 200 files.

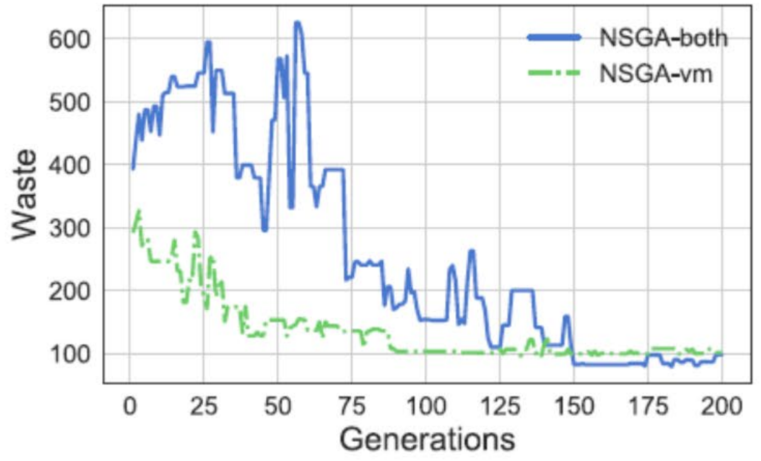


(f) Experiment size of 200 PMs and 200 files.

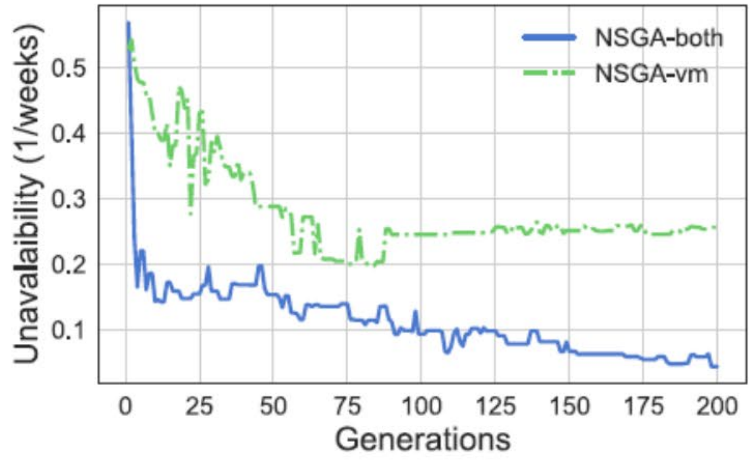
Experiment design, Result and discussions:



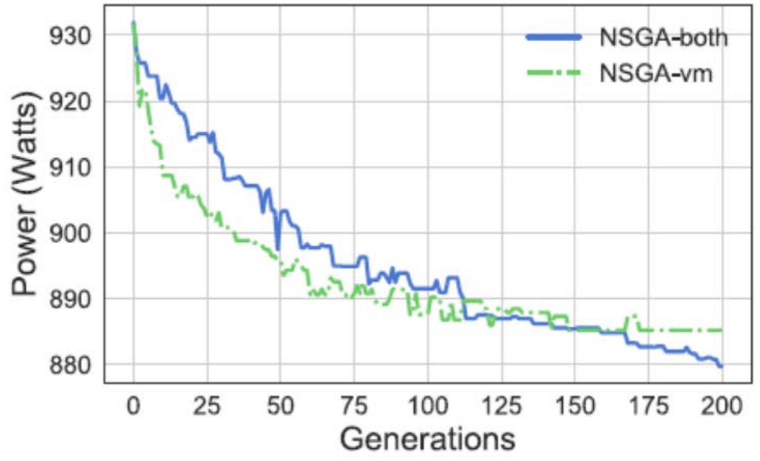
(a) Power consumption (50 PMs 25 files).



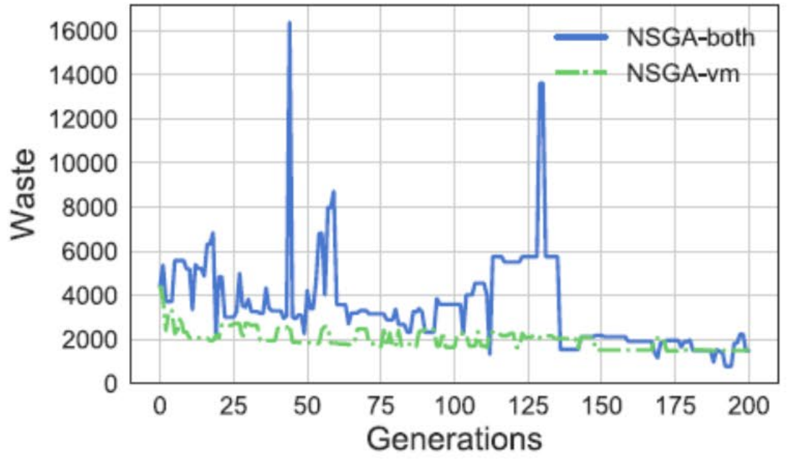
(b) Resource waste (50 PMs 25 files).



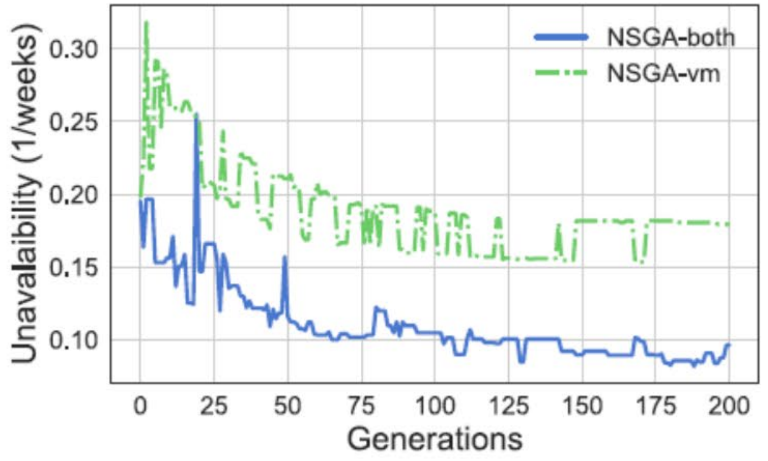
(c) File unavailability (50 PMs 25 files).



(d) Power consumption (200 PMs 200 files).



(e) Resource waste (200 PMs 200 files).



(f) File unavailability (200 PMs 200 files).

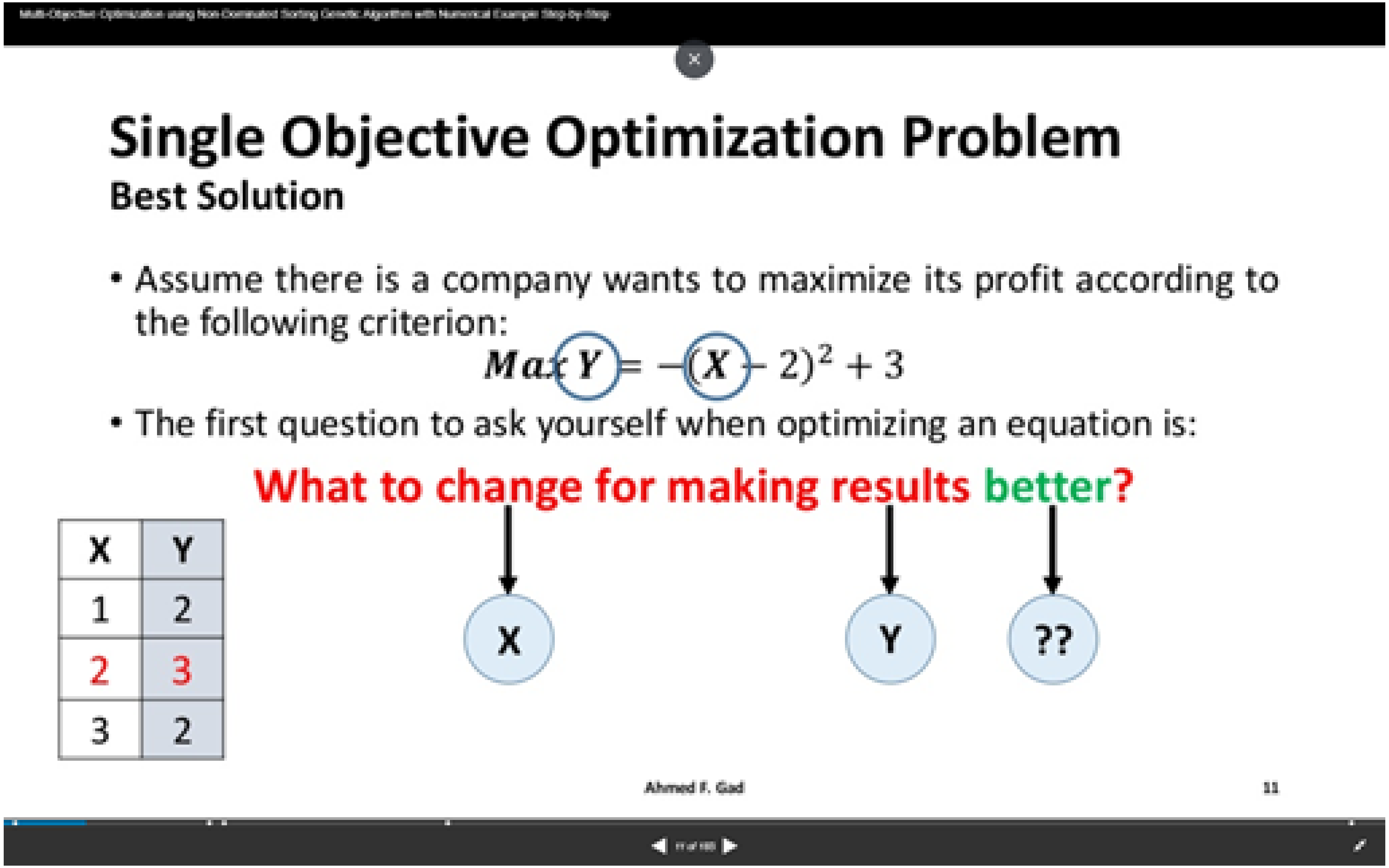
Conclusion:

- The solution minimizes three objectives: power consumption of the physical machines, waste of physical resources, and file unavailability.

- It has implemented the solution with NSGA-II, using a twofold chromosome and crossover operator to address this multi-objective optimization problem.

- The benefits in power consumption were measured to have an overall improvement of 1.9 percent. It showed a better behavior with the resource waste and file navailability, obtaining average improvements of 170.38 and 407.41 percent, respectively.

Additional info about GA and NSGA-II



Single Objective Optimization Problem

Best Solution

- Assume there is a company wants to maximize its profit according to the following criterion:

$$Max\ Y = -(X - 2)^2 + 3$$

- The first question to ask yourself when optimizing an equation is:

What to change for making results better?

X	Y
1	2
2	3
3	2

Single Objective Optimization Problem

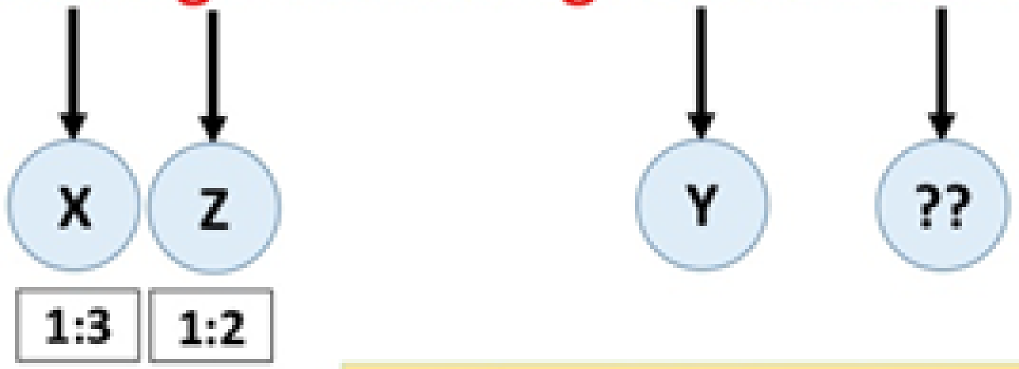
More Challenges

Add another **variable Z**

$$Max\ Y = Z^3 - (X-2)^2 + 3$$

What to change for making results better?

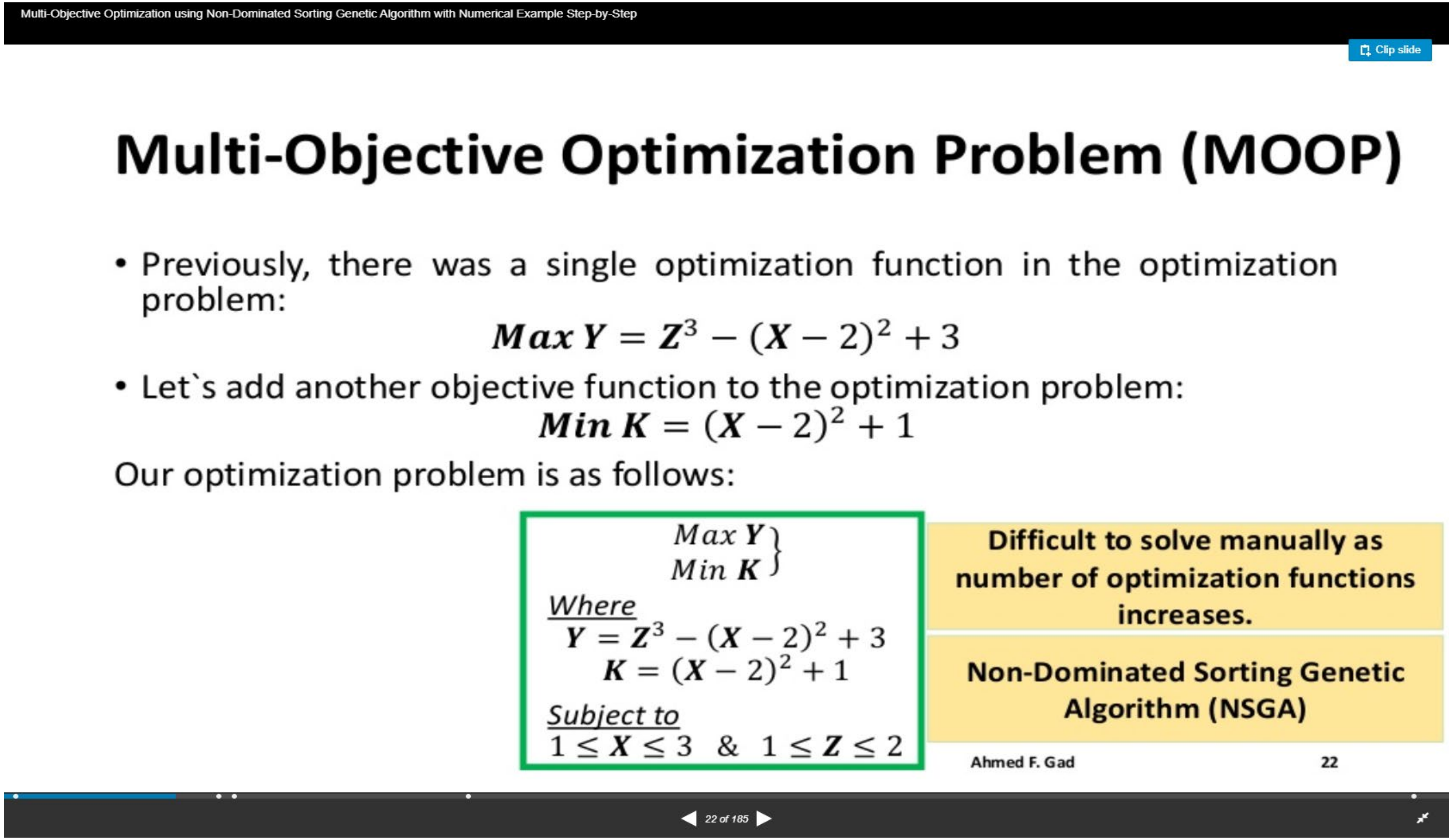
X	Z	Y
1	1	3
1	2	10
2	1	4
2	2	11
3	1	3
3	2	10



More Challenges

Unbounded values for input variables.

More objective functions.


Multi-Objective Optimization using Non-Dominated Sorting Genetic Algorithm with Numerical Example Step-by-Step
Clip slide
22 of 185

Multi-Objective Optimization Problem (MOOP)

- Previously, there was a single optimization function in the optimization problem:

$$\boldsymbol{Max\,Y} = \boldsymbol{Z}^3 - (\boldsymbol{X} - 2)^2 + 3$$

- Let`s add another objective function to the optimization problem:

$$\boldsymbol{Min\,K} = (\boldsymbol{X} - 2)^2 + 1$$

Our optimization problem is as follows:

$$\left.\begin{matrix} Max\ \boldsymbol{Y} \\ Min\ \boldsymbol{K} \end{matrix}\right\}$$

<u>*Where*</u>

$$\boldsymbol{Y} = \boldsymbol{Z}^3 - (\boldsymbol{X} - 2)^2 + 3$$
$$\boldsymbol{K} = (\boldsymbol{X} - 2)^2 + 1$$

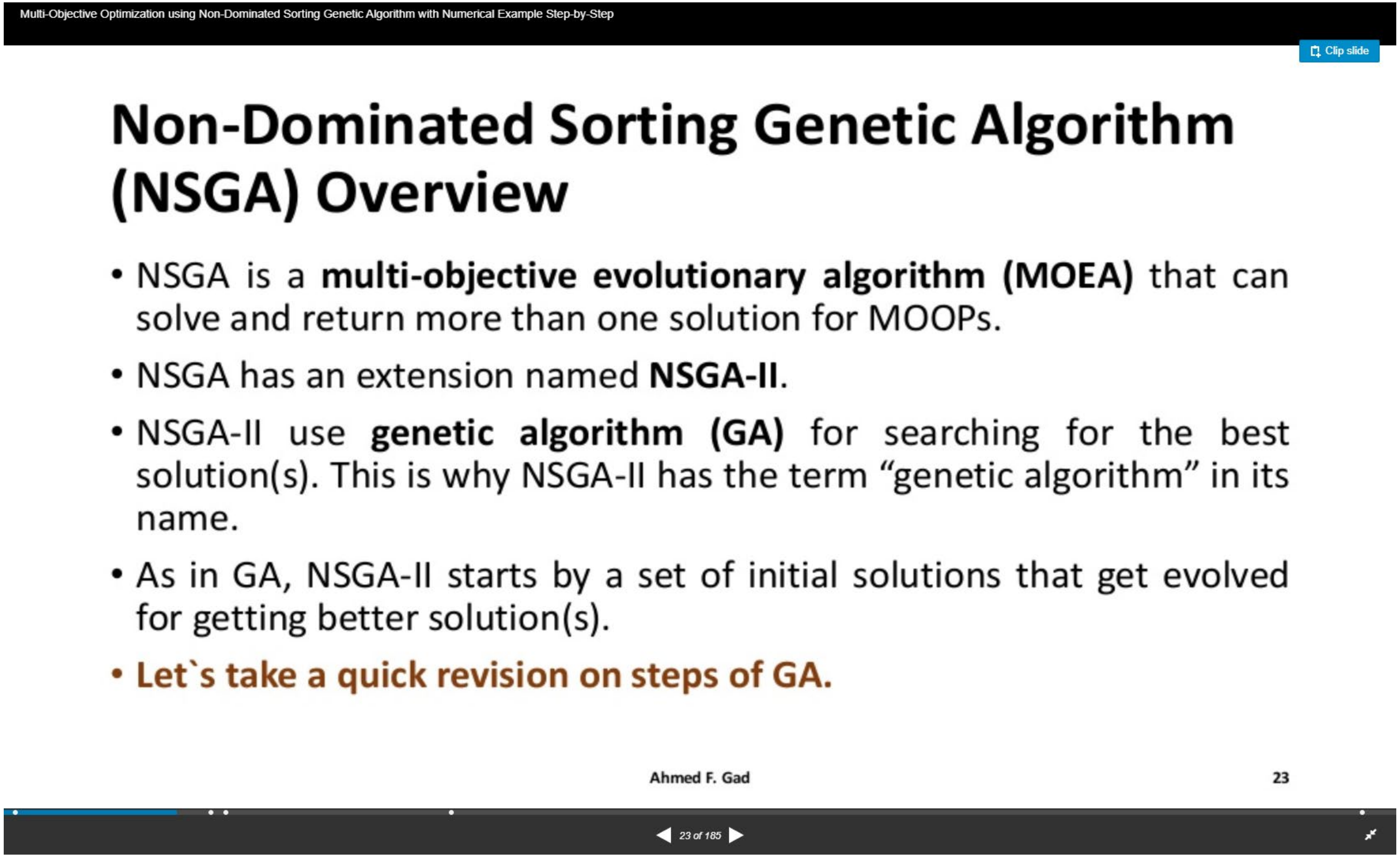
<u>*Subject to*</u>

$$1 \le \boldsymbol{X} \le 3 \quad \& \quad 1 \le \boldsymbol{Z} \le 2$$

Difficult to solve manually as number of optimization functions increases.

Non-Dominated Sorting Genetic Algorithm (NSGA)

Ahmed F. Gad



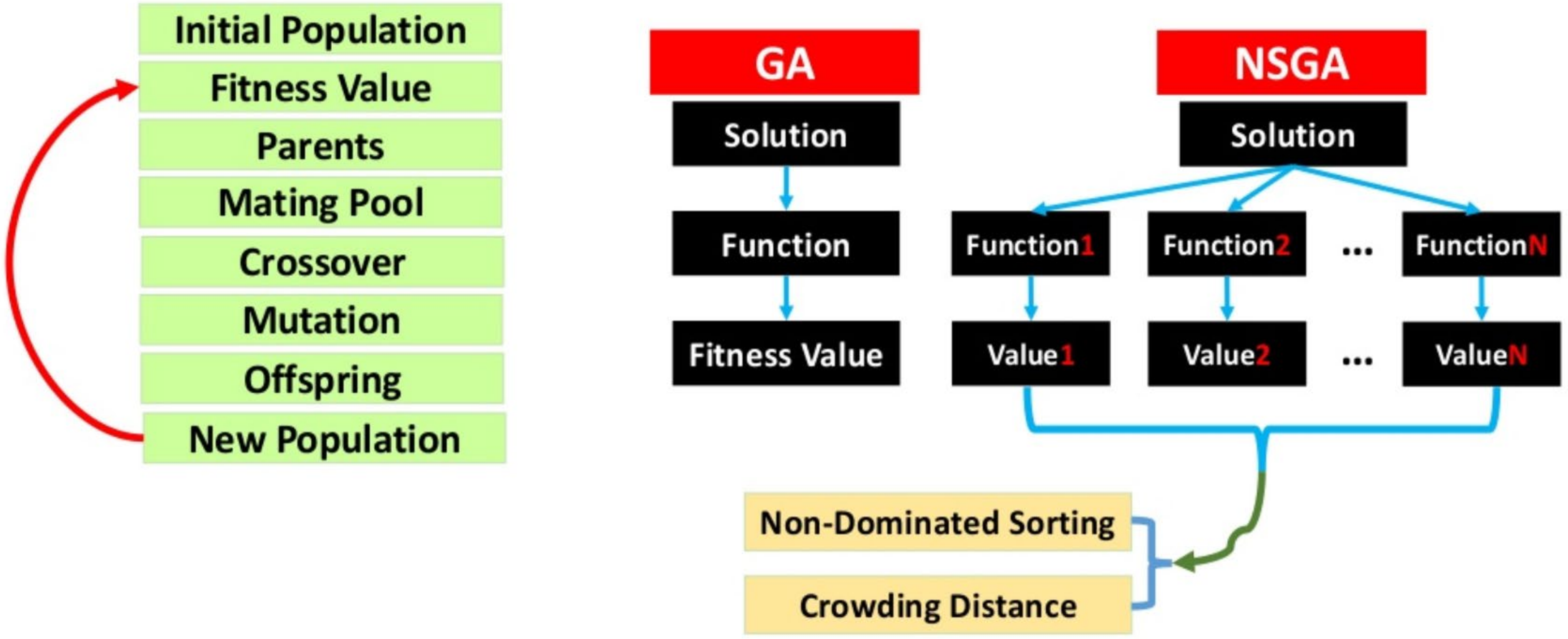
Non-Dominated Sorting Genetic Algorithm (NSGA) Overview

- NSGA is a **multi-objective evolutionary algorithm (MOEA)** that can solve and return more than one solution for MOOPs.
- NSGA has an extension named **NSGA-II**.
- NSGA-II use **genetic algorithm (GA)** for searching for the best solution(s). This is why NSGA-II has the term "genetic algorithm" in its name.
- As in GA, NSGA-II starts by a set of initial solutions that get evolved for getting better solution(s).
- **Let`s take a quick revision on steps of GA.**

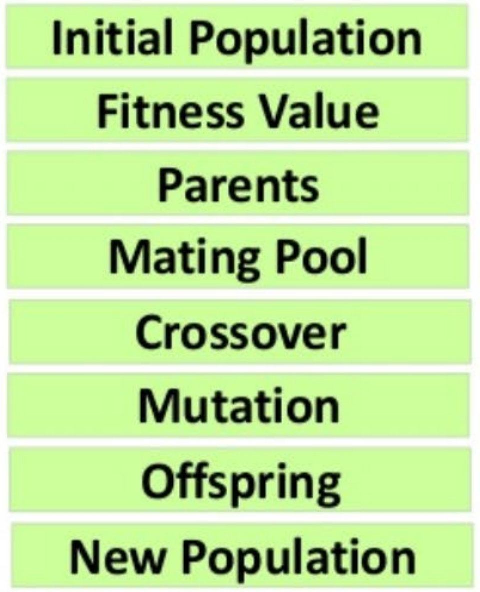
Clip slide

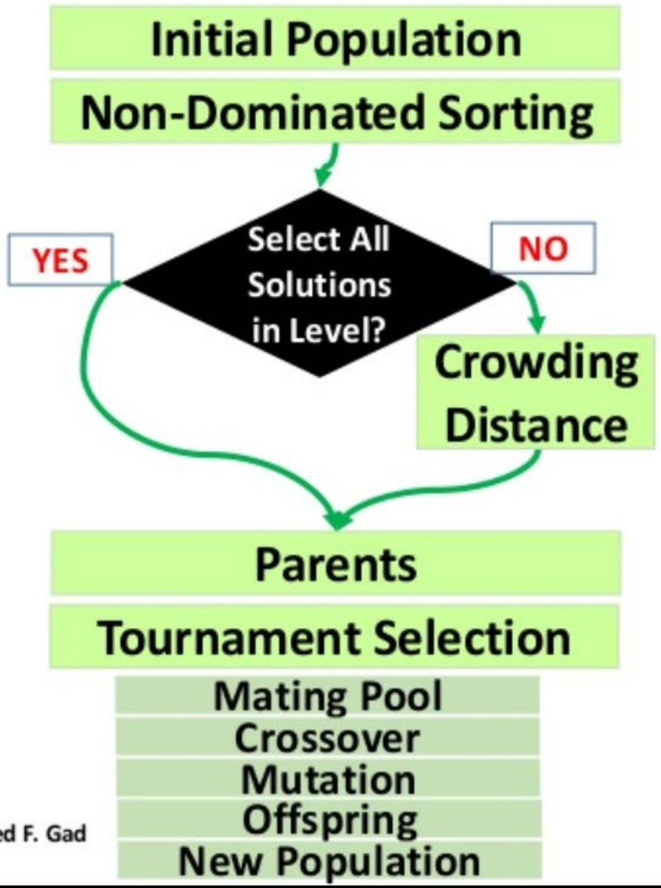
Genetic Algorithm (GA) Steps

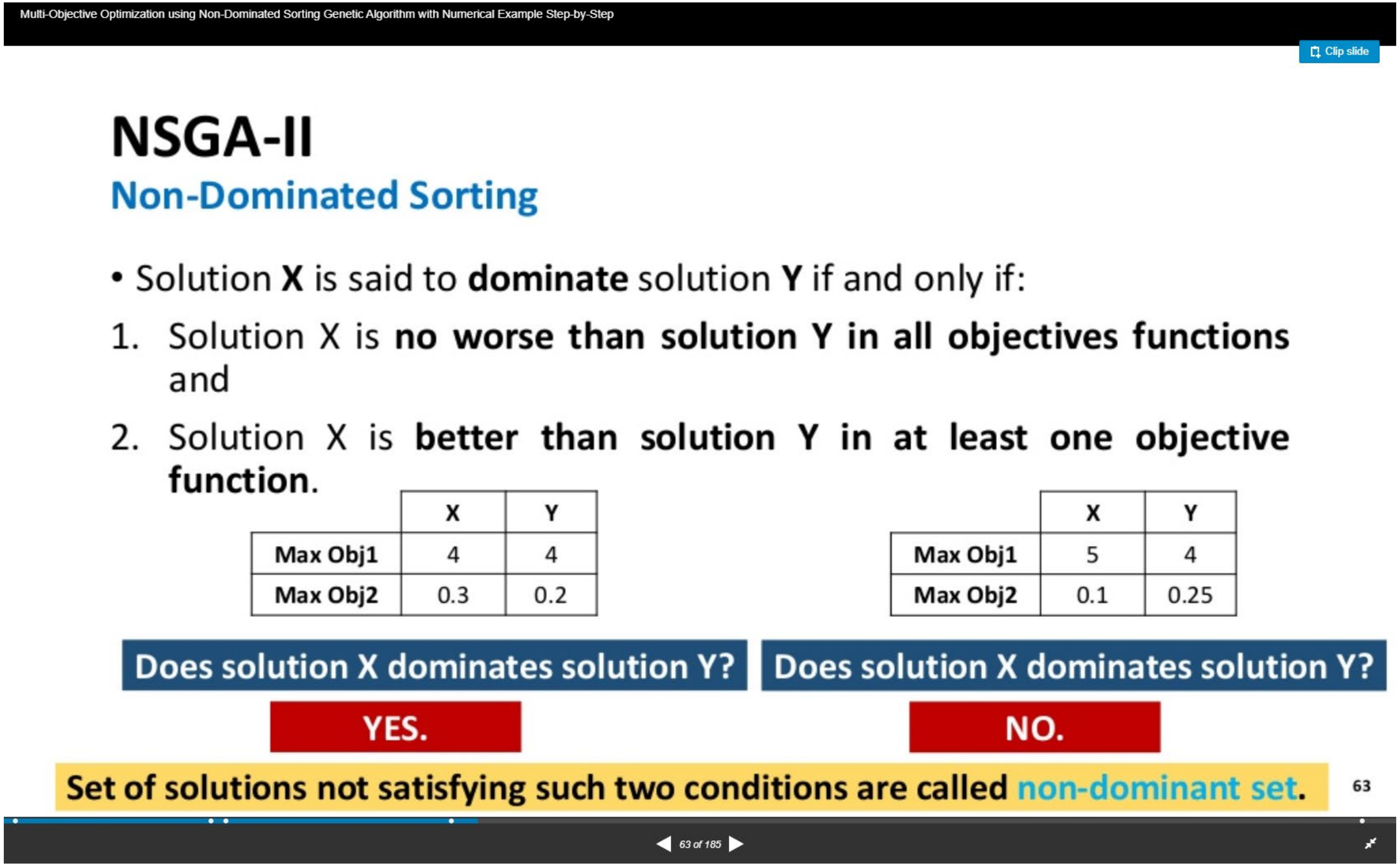
GA vs. NSGA-II



GA Vs. NSGA-II


Initial Population
Fitness Value
Parents
Mating Pool
Crossover
Mutation
Offspring
New Population


Initial Population
Non-Dominated Sorting
Select All Solutions in Level?
YES
NO
Crowding Distance
Parents
Tournament Selection
Mating Pool
Crossover
Mutation
Offspring
New Population



Clip slide

NSGA-II

Non-Dominated Sorting

- Solution **X** is said to **dominate** solution **Y** if and only if:

1. Solution X is **no worse than solution Y in all objectives functions** and
2. Solution X is **better than solution Y in at least one objective function**.

	X	Y
Max Obj1	4	4
Max Obj2	0.3	0.2

Does solution X dominates solution Y?

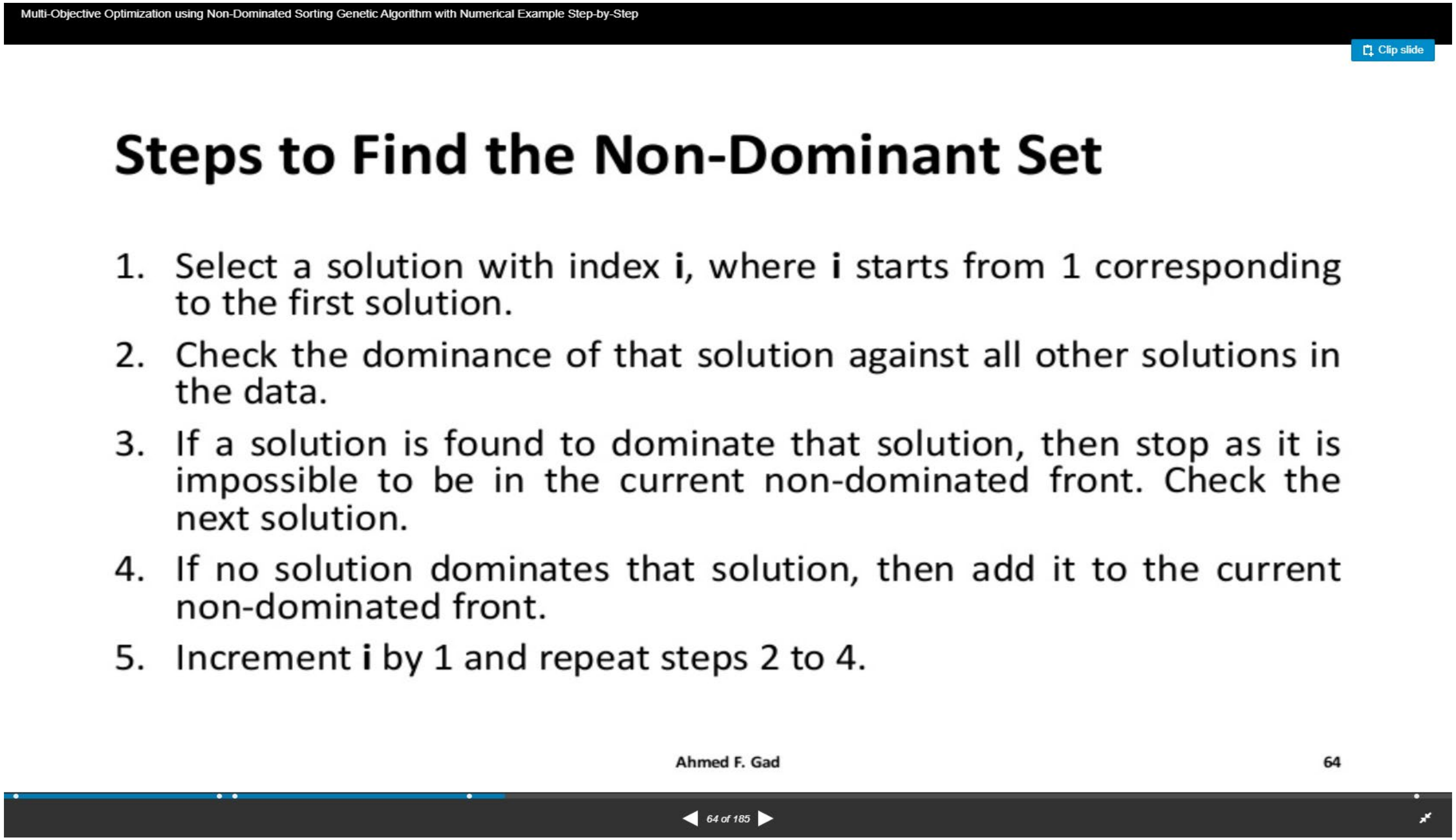
YES.

	X	Y
Max Obj1	5	4
Max Obj2	0.1	0.25

Does solution X dominates solution Y?

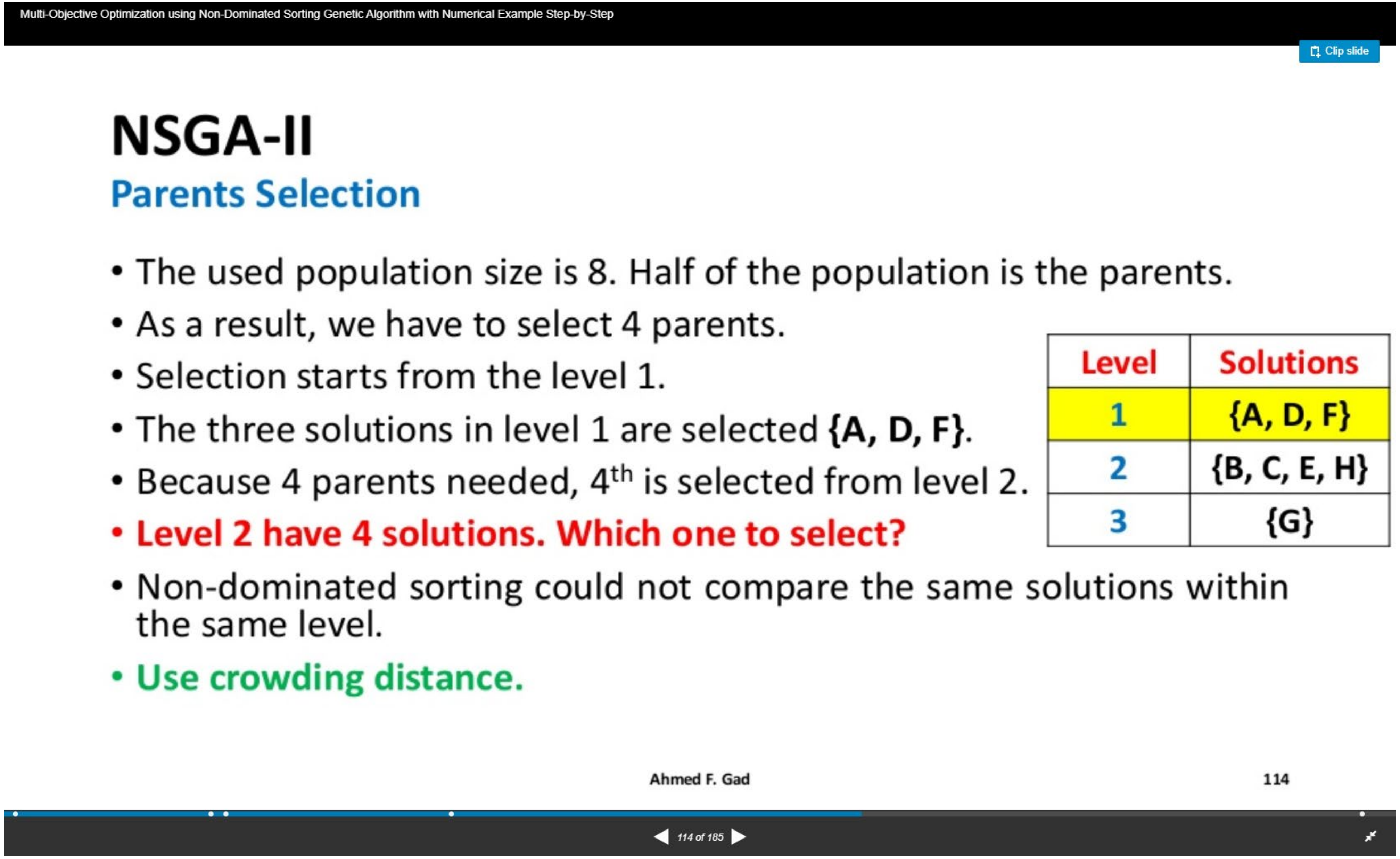
NO.

Set of solutions not satisfying such two conditions are called non-dominant set.



Steps to Find the Non-Dominant Set

1. Select a solution with index **i**, where **i** starts from 1 corresponding to the first solution.
2. Check the dominance of that solution against all other solutions in the data.
3. If a solution is found to dominate that solution, then stop as it is impossible to be in the current non-dominated front. Check the next solution.
4. If no solution dominates that solution, then add it to the current non-dominated front.
5. Increment **i** by 1 and repeat steps 2 to 4.

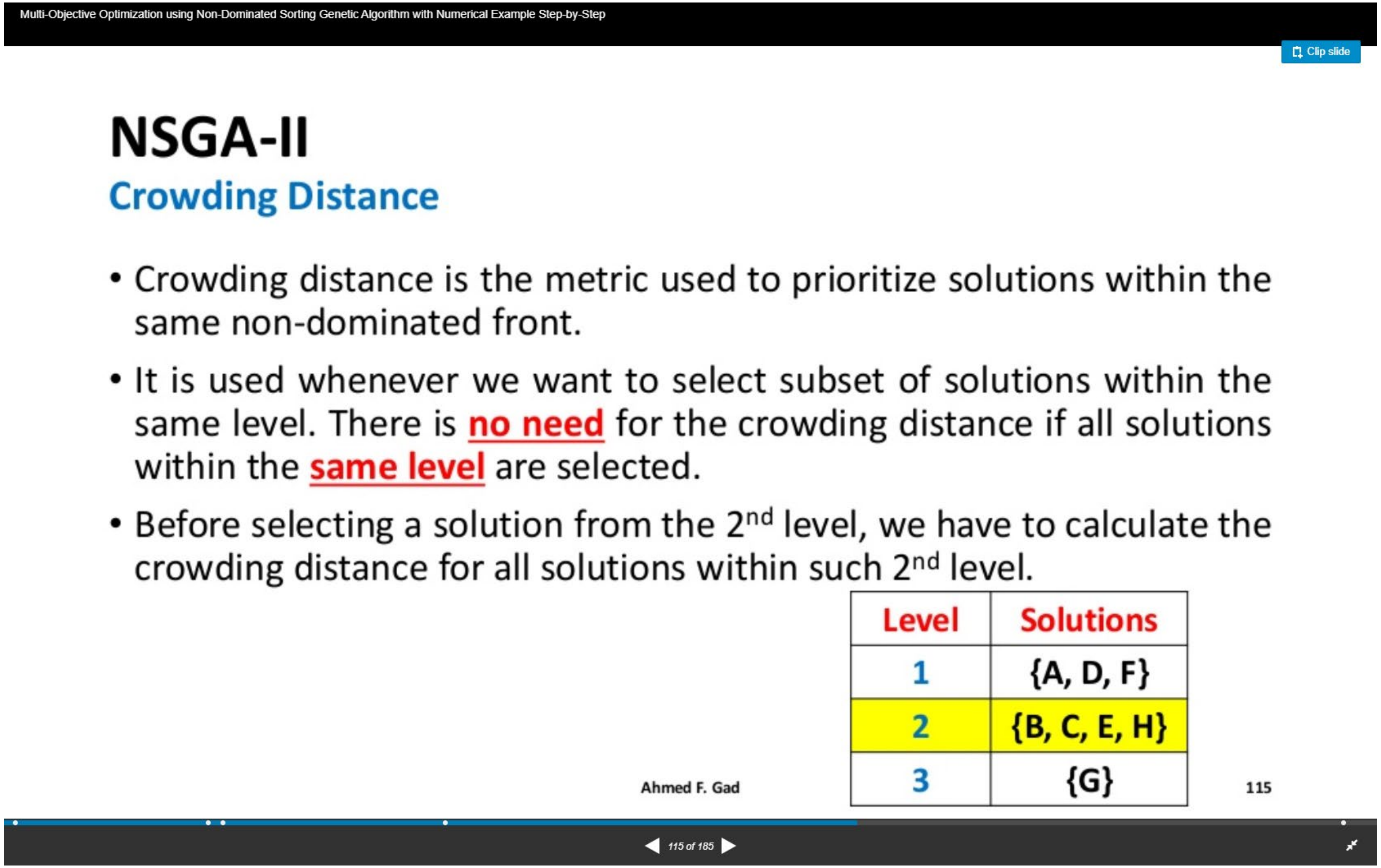


NSGA-II

Parents Selection

- The used population size is 8. Half of the population is the parents.
- As a result, we have to select 4 parents.
- Selection starts from the level 1.
- The three solutions in level 1 are selected **{A, D, F}**.
- Because 4 parents needed, 4th is selected from level 2.
- **Level 2 have 4 solutions. Which one to select?**
- Non-dominated sorting could not compare the same solutions within the same level.
- **Use crowding distance.**

Level	Solutions
1	{A, D, F}
2	{B, C, E, H}
3	{G}



Clip slide

NSGA-II

Crowding Distance

- Crowding distance is the metric used to prioritize solutions within the same non-dominated front.
- It is used whenever we want to select subset of solutions within the same level. There is **no need** for the crowding distance if all solutions within the **same level** are selected.
- Before selecting a solution from the 2nd level, we have to calculate the crowding distance for all solutions within such 2nd level.

Level	Solutions
1	{A, D, F}
2	{B, C, E, H}
3	{G}

Clip slide

NSGA-II

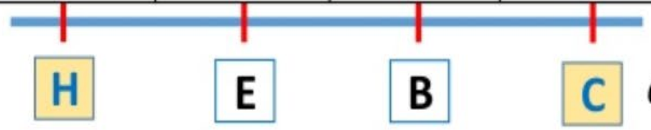
Crowding Distance – Step 3

$$d_m^n = \frac{S_m^{n+1} - S_m^{n-1}}{O_m^{max} - O_m^{min}}$$

Step 3: Crowding distance for solution E (n=3) according to the feedback objective (m=2).

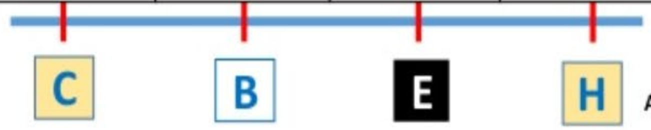
S_m^{n+1}	S_2^4	4.6
S_m^{n-1}	S_2^2	4.4
O_m^{max}	O_2^{max}	5
O_m^{min}	O_2^{min}	0

C. Distance	∞	0.4	0.1	∞
Cost $	25	55	60	65



$$d_2^3 = \frac{4.6 - 4.4}{5 - 0} = \frac{0.2}{5} = 0.04$$

C. Distance	∞	0.2	0.04	∞
Feedback	3.5	4.4	4.5	4.6



Clip slide

NSGA-II

Crowding Distance – Step 4

Step 4: Take the summation of the calculated crowding distances for all objectives.

ID	C. Distance Cost $	C. Distance Feedback	Summation
B	0.1	0.2	0.3
C	∞	∞	∞
E	0.4	0.04	0.44
H	∞	∞	∞

NSGA-II

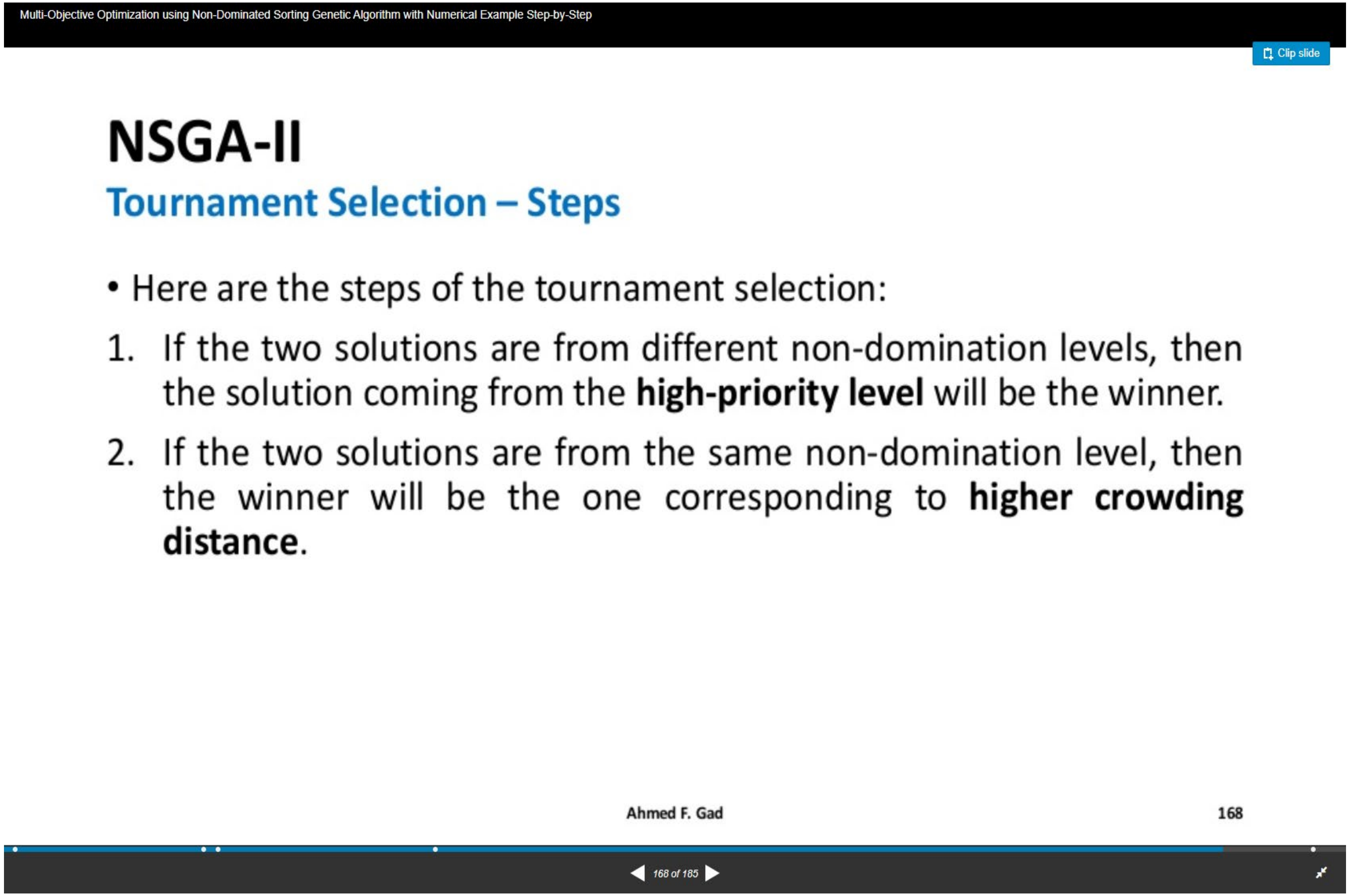
Crowding Distance – Step 5

Step 5: Sort them in descending order according to summation of crowding distance and select the solutions from highest to lowest crowding distance.

ID	Summation
B	0.3
C	∞
E	0.44
H	∞

Sort

ID	Summation
C	∞
H	∞
E	0.44
B	0.3

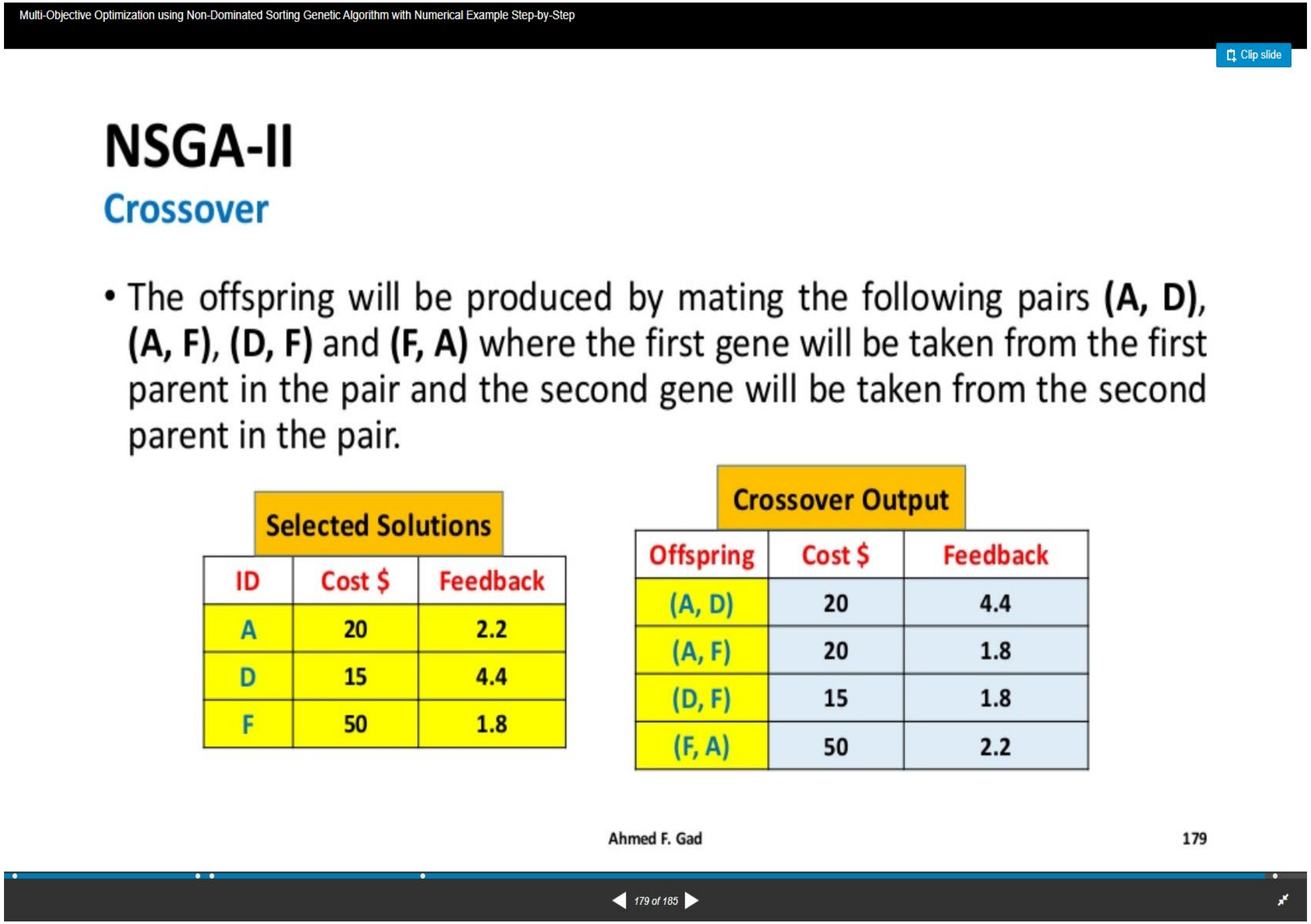


NSGA-II

Tournament Selection – Steps

- Here are the steps of the tournament selection:

1. If the two solutions are from different non-domination levels, then the solution coming from the **high-priority level** will be the winner.
2. If the two solutions are from the same non-domination level, then the winner will be the one corresponding to **higher crowding distance**.



Multi-Objective Optimization using Non-Dominated Sorting Genetic Algorithm with Numerical Example Step-by-Step

Clip slide

NSGA-II

Crossover

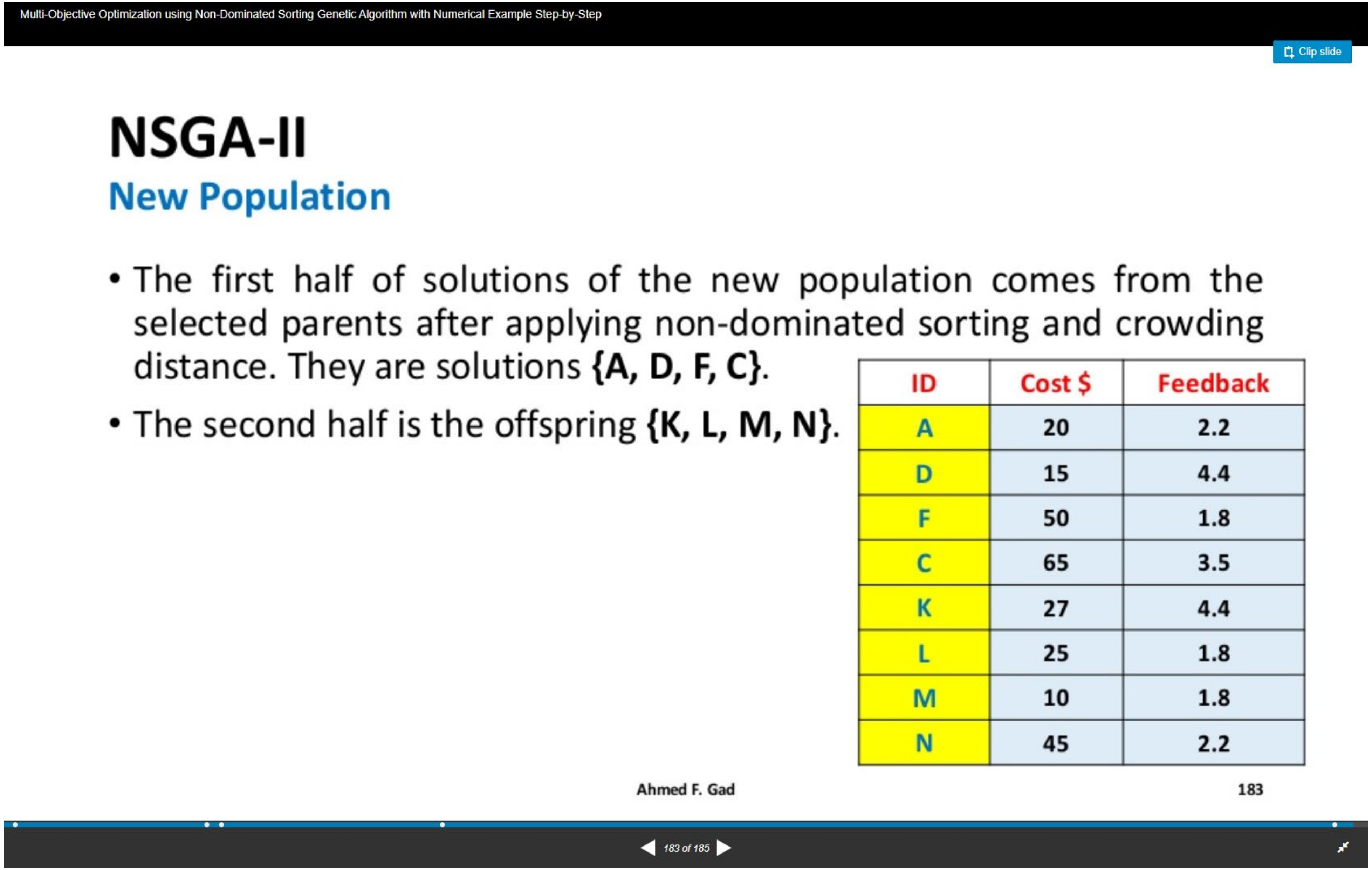
- The offspring will be produced by mating the following pairs **(A, D)**, **(A, F)**, **(D, F)** and **(F, A)** where the first gene will be taken from the first parent in the pair and the second gene will be taken from the second parent in the pair.

Selected Solutions

ID	Cost $	Feedback
A	20	2.2
D	15	4.4
F	50	1.8

Crossover Output

Offspring	Cost $	Feedback
(A, D)	20	4.4
(A, F)	20	1.8
(D, F)	15	1.8
(F, A)	50	2.2



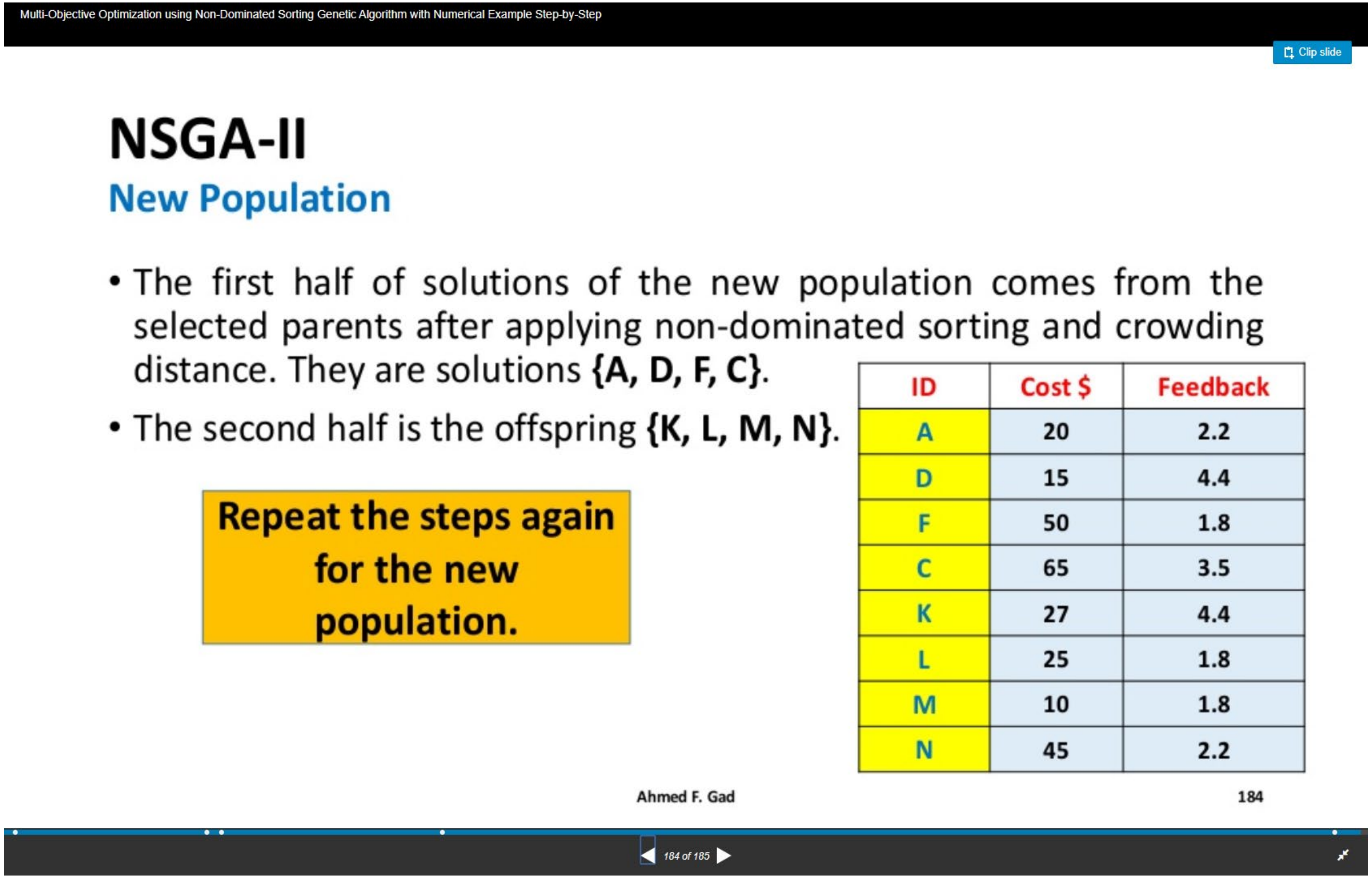
Clip slide

NSGA-II

New Population

- The first half of solutions of the new population comes from the selected parents after applying non-dominated sorting and crowding distance. They are solutions **{A, D, F, C}**.
- The second half is the offspring **{K, L, M, N}**.

ID	Cost $	Feedback
A	20	2.2
D	15	4.4
F	50	1.8
C	65	3.5
K	27	4.4
L	25	1.8
M	10	1.8
N	45	2.2



NSGA-II

New Population

- The first half of solutions of the new population comes from the selected parents after applying non-dominated sorting and crowding distance. They are solutions **{A, D, F, C}**.
- The second half is the offspring **{K, L, M, N}**.

Repeat the steps again for the new population.

ID	Cost $	Feedback
A	20	2.2
D	15	4.4
F	50	1.8
C	65	3.5
K	27	4.4
L	25	1.8
M	10	1.8
N	45	2.2