Optimization of File Compilation by Meta-heuristic and Mathematical Techniques

Eshu Patel 20007026
Suvayan Nath 200107088

Introduction

- We frame the optimization problem for file compilation on servers, and discuss the constraints and objective function.
- A meta-heuristic approach using genetic algorithm (GA)is proposed to obtain a feasible solution.Both single-objective GA and multi-objective GA (MOGA) are explored.
- A mathematical formulation (MINLP) is further shown to obtain the optimal solution to the problem.

Problem Definition

- The problem of file compilation involves compiling multiple files on multiple servers while considering file dependencies, with the constraint that certain files require their dependencies to be compiled and available on the same server before compilation can begin. Similar to JSSP.
- Every file has a replication time after compilation, beyond which it is accessible to all servers.
- Deadlock can occur when files are blocked, unable to proceed because they are waiting for resources held by other files in a circular dependency. Deadlocks can cause system crashes and reduced performance, particularly in multi-process systems.
- The replication time constraint means feasible sequences can often result in servers undergoing large idle periods while waiting for access to a dependency, leading to deadline violations.

Meta-Heuristic Approach

- We extensively alter the original GA, and formulate a set of decision variables or chromosome for the problem at hand. A step-by-step approach is described for our version of GA, suited for permutations.
- To represent our encoding scheme, we take up a case of compilation of 6 files on 2 servers. Let Fi define the ith file, for i ε {1..6}. We express a solution S as

$$S = [F1\ F3\ F6\ ||\ F2\ F4\ F5]$$

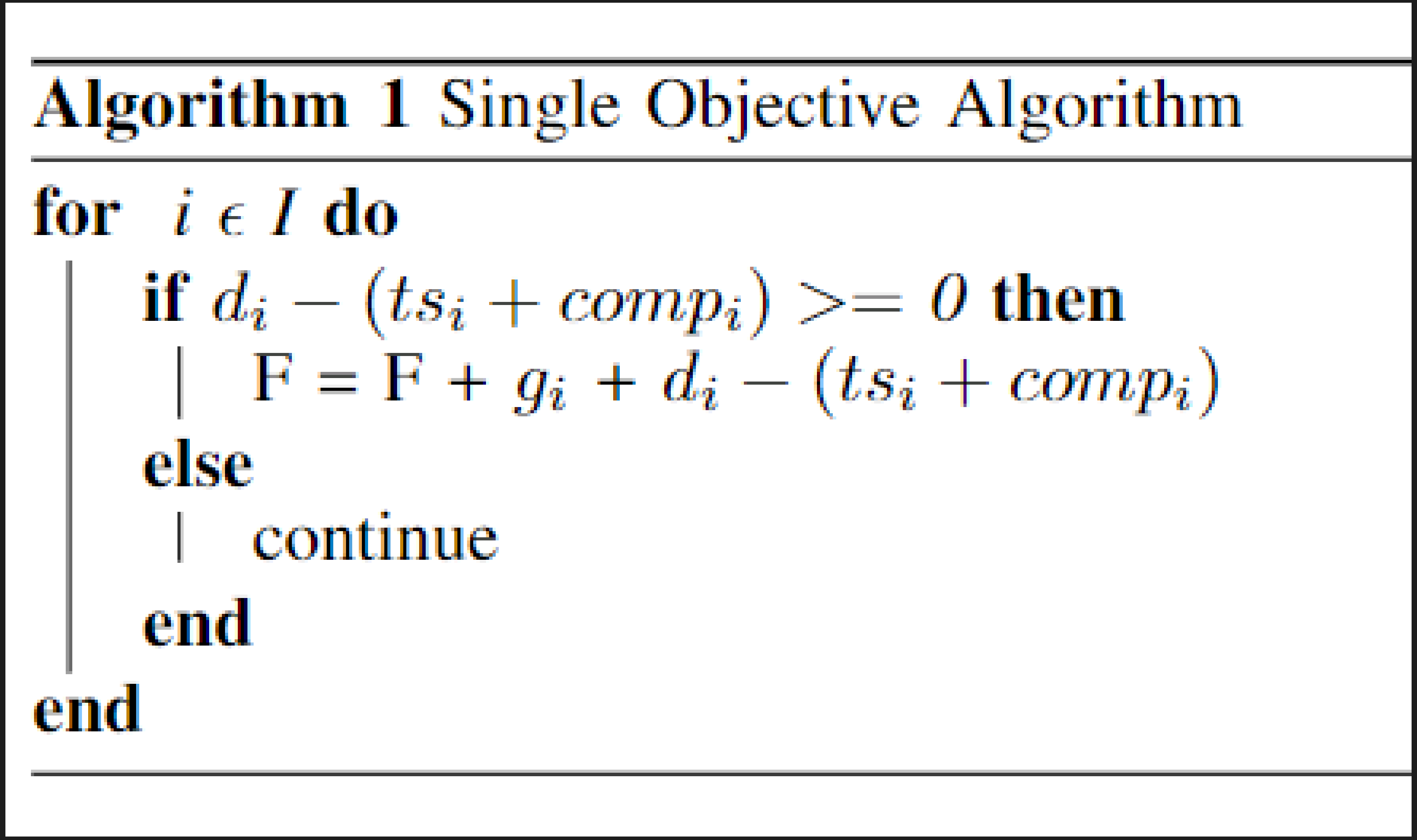
- where the first set of files are compiled in the 1st server in that sequence, and so on, with ” // ” serving as a partition between compilation lists of every server.
- In code, S is represented as follows: S = [1 3 6 0 2 4 5] The numbers are synonymous with file numbers, and ”0” is our partition symbol.

Meta-Heuristic Approach

- We use LOX (Linear order crossover) for cross-over in GA. The reason for LOX variant over standard cross- overs like Order Crossover (OX) is that, OX and similar operators treat chromosomes as circular, but this causes destruction of the priority structure for our chromosome. A feasible solution could become a infeasible one very easily, which we try to minimize. In other words, we preserve the high and low priorities in our chromosome parents, and pass it on to the offspring.

- Mutation is the process of diversifying offspring. Again, random changes cannot be done to our chromosome, so we opt for a simple binary inversion. We choose two positions and swap the digits present therein. We only perform this once, as multiple inversions can lead to worsening of the solution.

Objective Function



Algorithm 1 Single Objective Algorithm

```
for $i \in I$ do
    if $d_i - (ts_i + comp_i) >= 0$ then
        F = F + $g_i + d_i - (ts_i + comp_i)$
    else
        continue
    end
end
```

We combine objectives for both file completion within deadlines, and tardiness measure into a single objective. Let there be I files, each with deadlines di, start times tsi, compilation time compi and goals gi.The objective F, initialized with 0 can be defined as the following pseudo-code.

Starting Time and deadlock

We use an iterative approach to figure out starting times or un-feasibility from the chromosome. We iterate through every server to see if they are able to compile the next file in their sequence. Every server maintains a local time. If in a epoch, we find no server is able to compile and the replication periods are over across other servers, we detect a deadlock and apply penalty.

Mathematical Formulation

The mathematical formulation of this file compilation problem is inspired by mixed integer linear programming(MILP) formulation of the classical Job shop scheduling problem (JSSP). The formulation we are going to discuss is for single objective mixed integer non-linear problem (MINLP).

Objective - We have to compile all the files before their deadlines and maximize the total points.

x_im is a assignment variable which is binary which is one if file i is to assigned on server m and zero otherwise. Another binary variable y_ii' is a sequencing variable which takes the value one when both files i and i' are assigned to the same server and file i' is to be compiled after file i, otherwise it will be zero. Start time of a file i is indicated by the continuous variable ts_i. comp_i is the compilation time, rep_i is the replication time of file i. dep_ii' is the binary variable which is one if $i_'$ is the dependency of i, zero otherwise. T is the list of target files.t_i is a binary variable if it is one it means file i is one of our target file.

Explanation

Objective(1) is to maximize the total points obtained by compiling all the target files before deadline. Constraint(2) ensures that start time of a file must be atleast comp_i times ahead of it's deadline, if this contraint gets violated this will mean that file i not compiled before its deadline. Any file can be compiled on only one server constraint(3) takes care of that, for a given i x_im will be one for any one m and zero for others. Constraint(4) ensures that the total time for which any server is compiling the files must be less than the latest deadline, if this constraint gets violated this will mean that there is at least one file which is not compiled before its deadline, this constraint tightens the LP relaxation of the problem. Constraint(5) is based on the fact that if the files i and i' are assigned to the same server then they must be compiled one after another. Constraint(6) is a sequencing constraint which makes sure that if binary variable dep_ii' is one then i' must be compiled after i.

$$max \sum_{i \epsilon I} \sum_{m \epsilon M} t_i(g_i + (d_i - (ts_i + c_i))) \qquad (1)$$

s.t.

$$ts_i \leq d_i - \sum_{m \epsilon M} c_i x_{im} \quad \forall i \epsilon I \qquad (2)$$

$$\sum_{m \epsilon M} x_{im} = 1 \quad \forall i \epsilon I \qquad (3)$$

$$\sum_{i \epsilon I} x_{im} p_i = \max_i d_i \qquad (4)$$

$$y_i + y_{i'} \geq x_i + x_{i'} - 1 \;\; \forall i, i' \epsilon I, i' \; > \; i, m \epsilon M \qquad (5)$$

$$ts'_i \geq ts_i + \sum_{m \epsilon M} comp_i x_{im} - U(1 - y_{ii'})$$

$$\forall i, i' \epsilon I, i \neq i' \qquad (6)$$

$$y_{ii'} + y_{i'i} \leq 1 \quad \forall i, i' \epsilon I, i' \; > \; i \qquad (7)$$

Explanation(continued..)

Constraint(7) is logical cut, it is based on the fact that if i and i' are assigned to the same server then either of y_ii' and y_i'i would be one, but if they are assigned on different machine then both y_ii' and y_i'i would be zero. Constraint(8) is also a logical relationship which makes sure that if the files i and i' are assigned to different server then the sequencing variables in this equation will be zero. Constraint(9) ensures that if i_' is the dependency of i and i is compiling on the same server on which i' has been compiled then start time of i must be finished time of i', but if i is compiling on some other server then start time of i must be after (finished time + replication time) of ii' . Constraint(10) is a obvious constraint which makes sure that start time can not be negative. Constraints(11) and (12) makes sure that x_im and y_ii' are binary variable.

$$y_{ii'} + y_{i'i} + x_{im} + x_{i'm'} \leq 2$$
$$\forall i, i' \in I, i > i', m, m' \in M, m \neq m' \quad (8)$$

$$ts_i \geq dep_{ii'}(ts_{i'} + comp_{i'} + x_{im}(1 - x_{i'm})rep_{i'}$$
$$\forall i, i' \in I, i > i', m \in M \quad (9)$$

$$ts_i \geq 0 \quad (10)$$

$$x_{im} \in \{0, 1\} \ \ \forall i \in \ I, m \in M \quad (11)$$

$$y_{ii'} \in \{0, 1\} \ \ \forall i, i' \in \ I, i \neq i' \quad (12)$$

Results

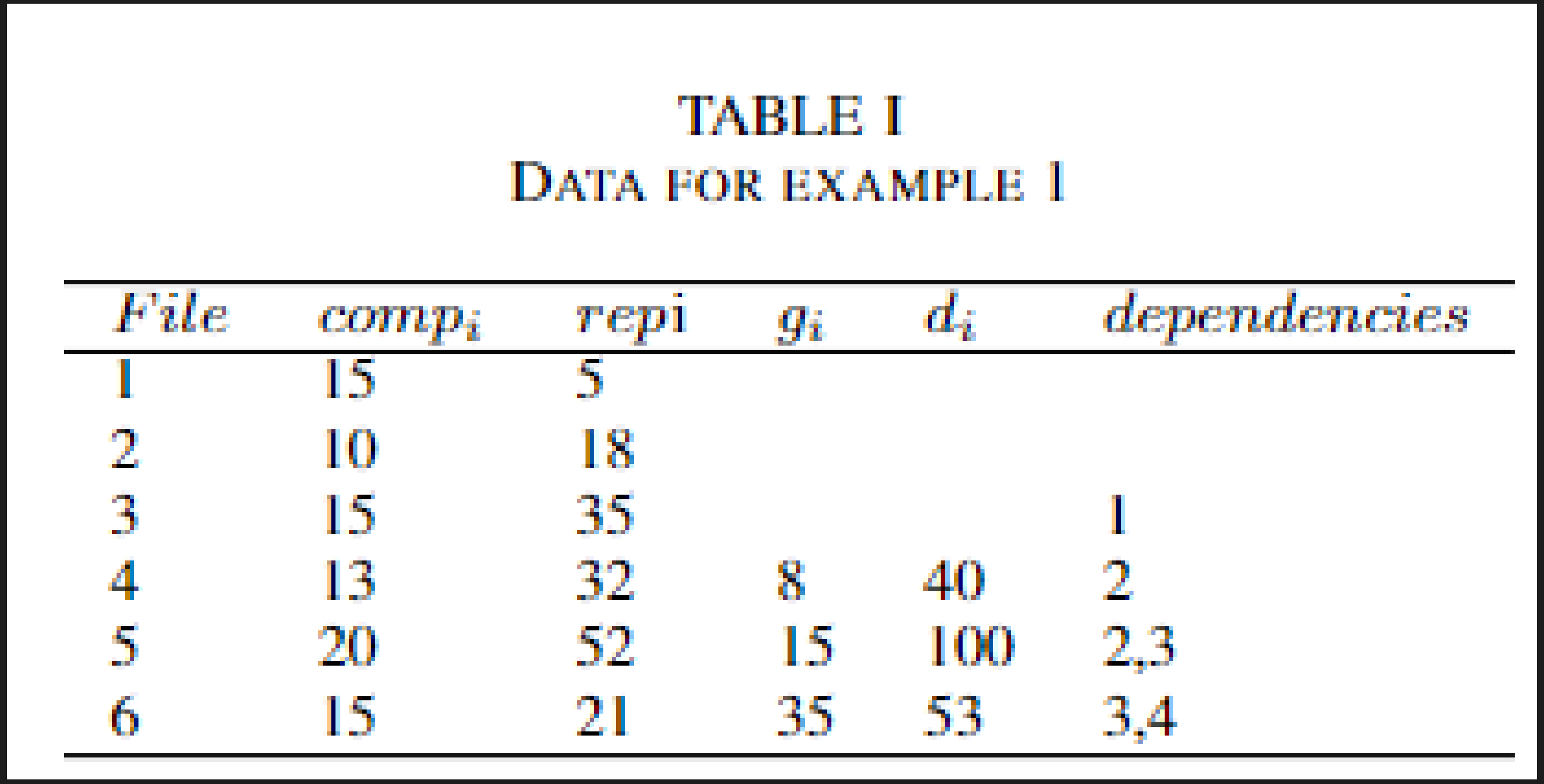


TABLE I
DATA FOR EXAMPLE 1

File	$comp_i$	*repi*	g_i	d_i	*dependencies*
1	15	5			
2	10	18			
3	15	35			1
4	13	32	8	40	2
5	20	52	15	100	2,3
6	15	21	35	53	3,4

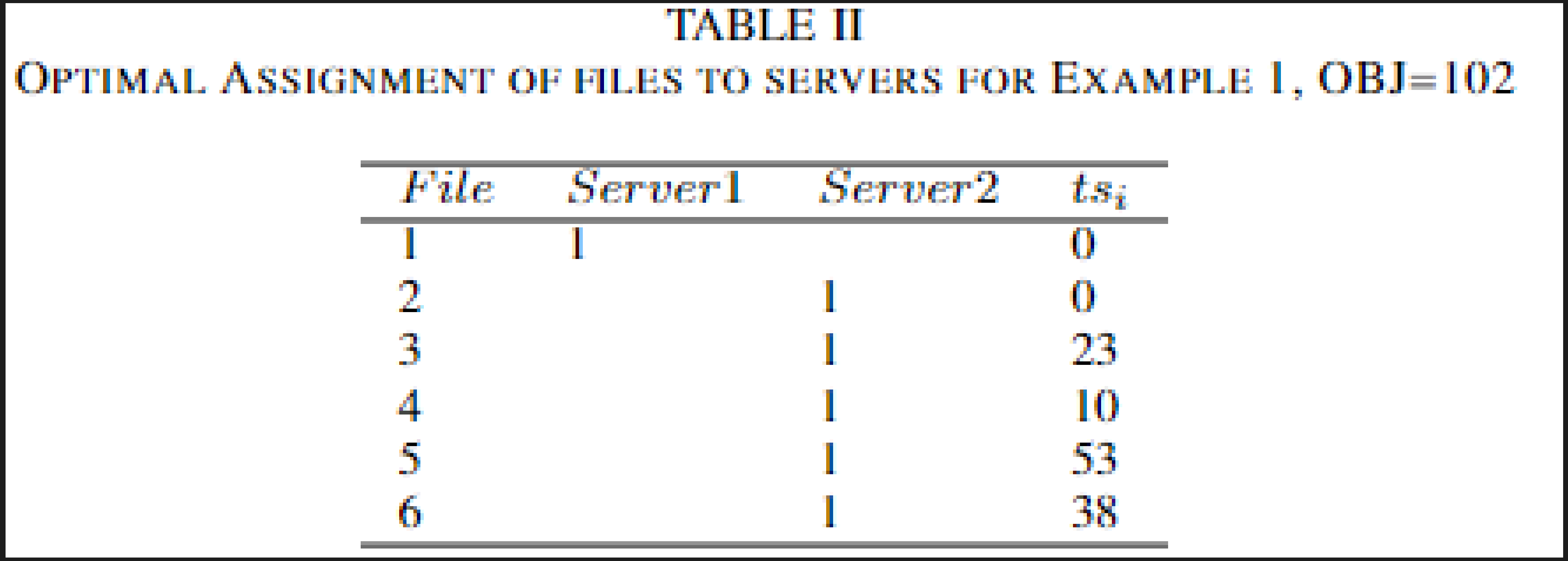
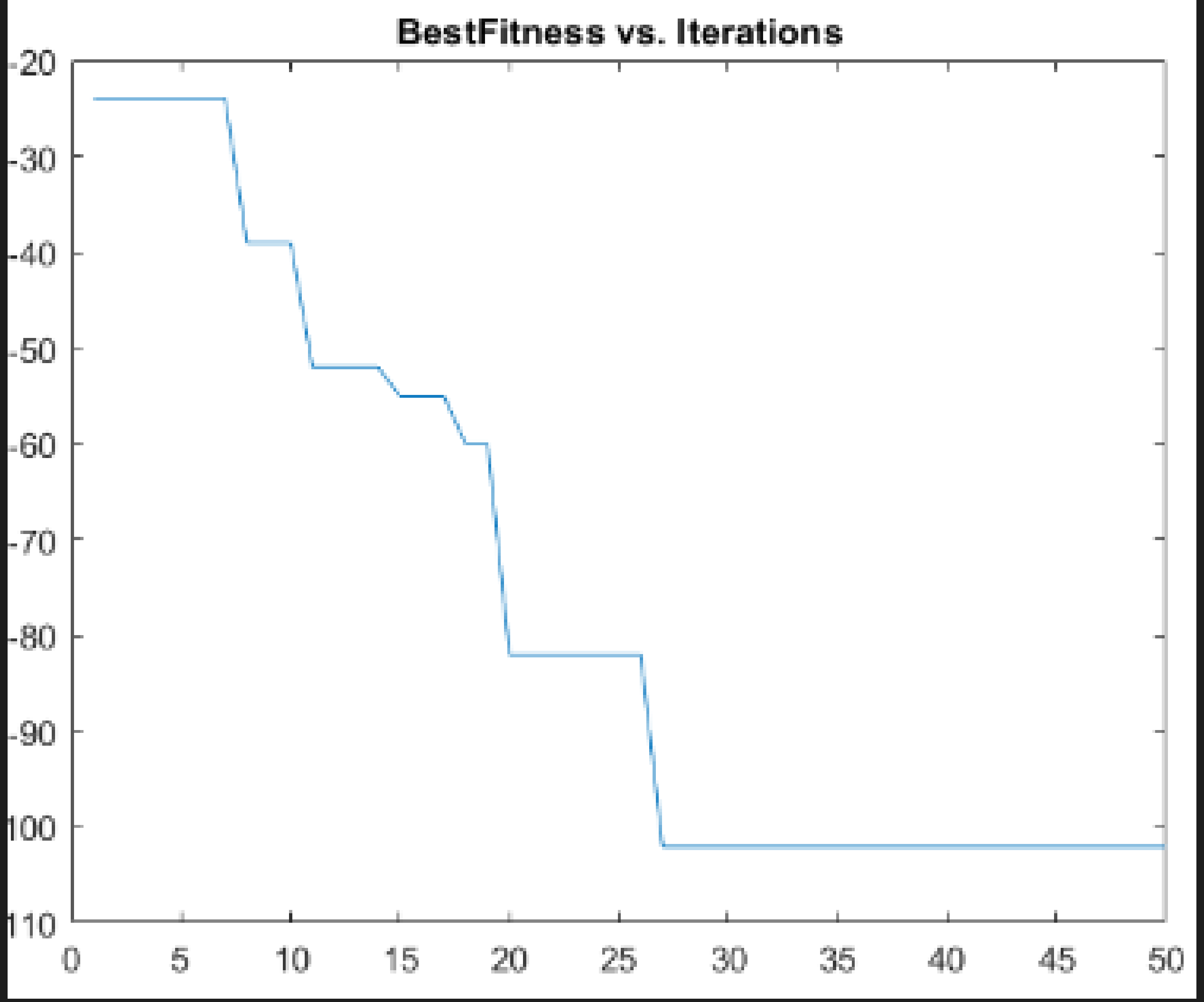


TABLE II
OPTIMAL ASSIGNMENT OF FILES TO SERVERS FOR EXAMPLE 1, OBJ=102

File	*Server1*	*Server2*	ts_i
1	1		0
2		1	0
3		1	23
4		1	10
5		1	53
6		1	38

Fig. 1. Convergence Plot for Single-Objective GA



Results

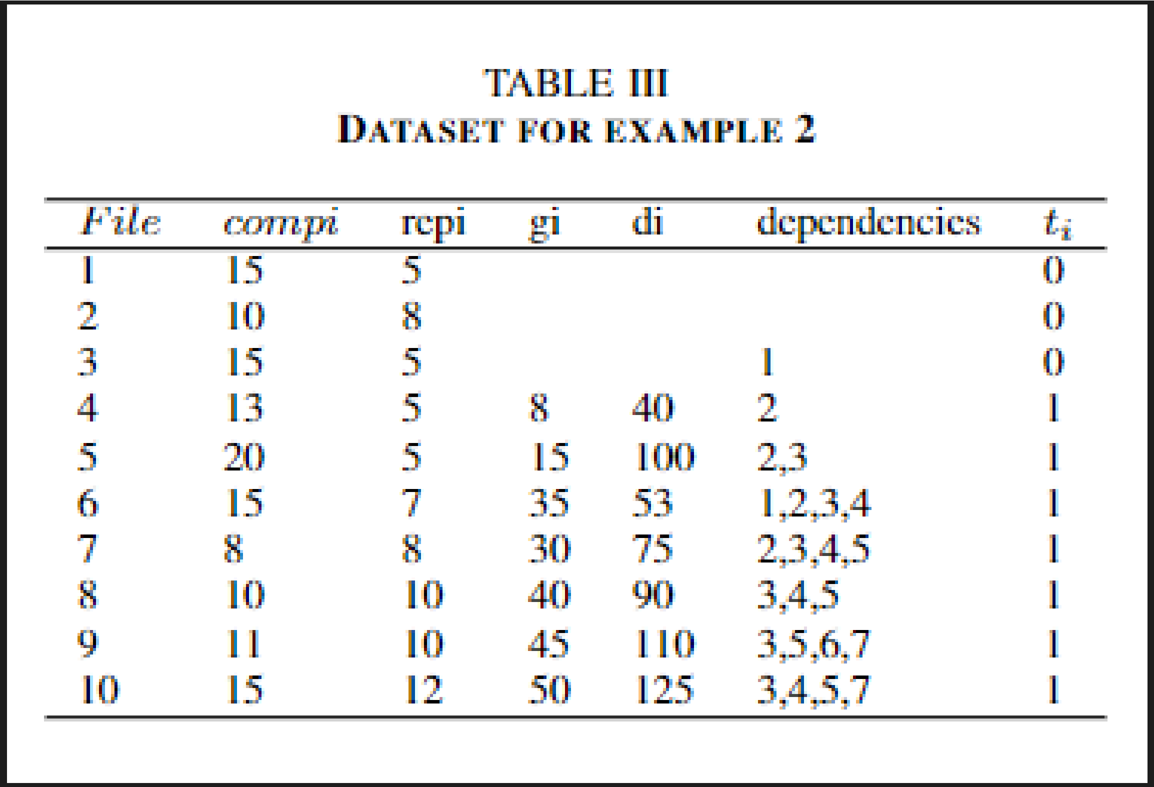


TABLE III
DATASET FOR EXAMPLE 2

File	*compi*	repi	gi	di	dependencies	t_i
1	15	5				0
2	10	8				0
3	15	5			1	0
4	13	5	8	40	2	1
5	20	5	15	100	2,3	1
6	15	7	35	53	1,2,3,4	1
7	8	8	30	75	2,3,4,5	1
8	10	10	40	90	3,4,5	1
9	11	10	45	110	3,5,6,7	1
10	15	12	50	125	3,4,5,7	1

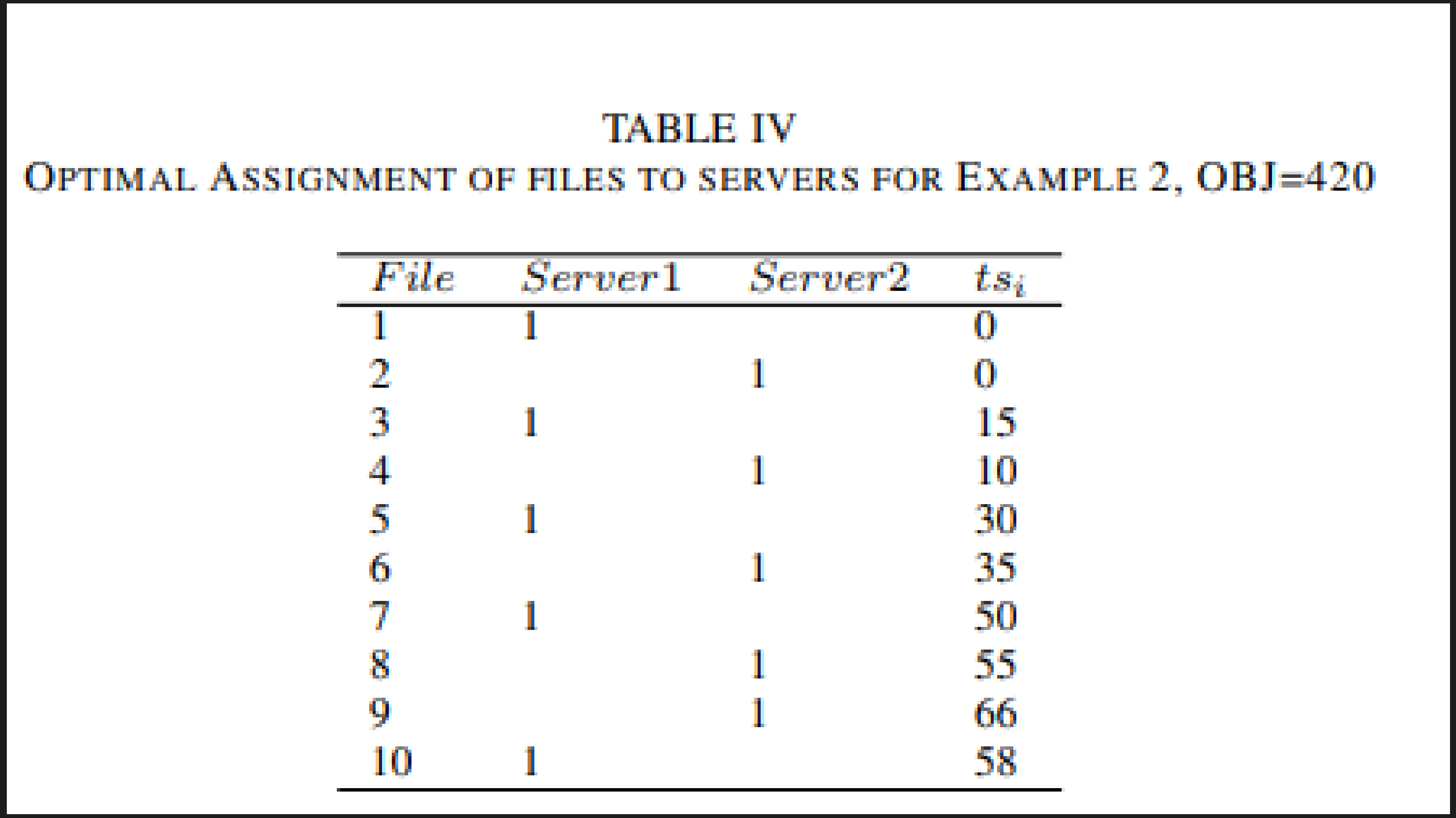
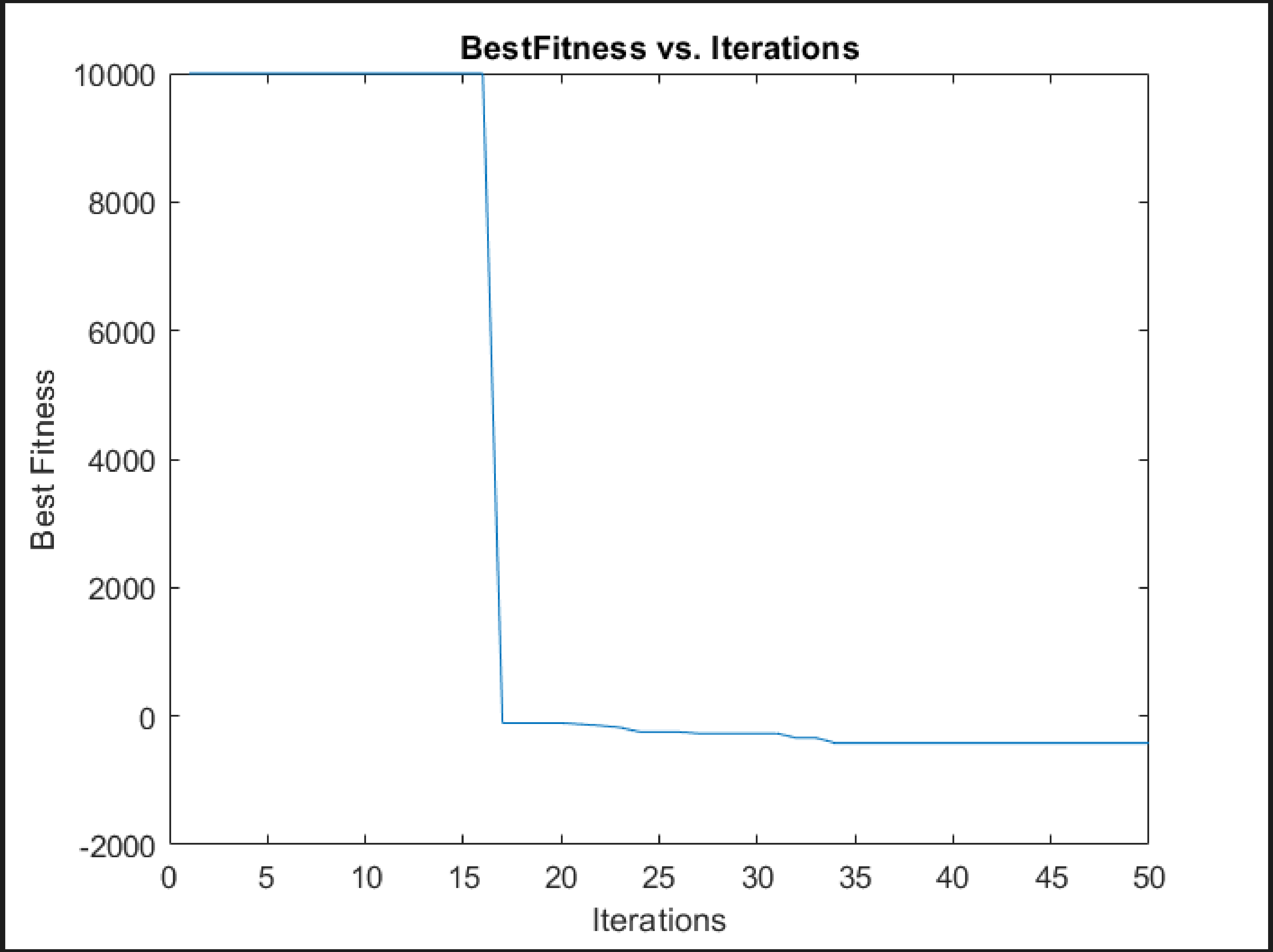


TABLE IV
OPTIMAL ASSIGNMENT OF FILES TO SERVERS FOR EXAMPLE 2, OBJ=420

File	*Server1*	*Server2*	ts_i
1	1		0
2		1	0
3	1		15
4		1	10
5	1		30
6		1	35
7	1		50
8		1	55
9		1	66
10	1		58



Conclusions

Our example establishes the supremacy of mathematical formulation over the meta-heuristic techniques like GA. But both meta heuristic techniques and mathematical formulations has their own advantages and disadvantages. We can solve any black box problem using metaheuristic techniques without even having its mathematical model while we can use mathematical formulations only if we have mathematical model for the problem. Mathematical formulations are computationally intensive but they guarantees the best solution(if formulated properly) while metaheuristic techniques does not give us any such guarantee so we need to test the problem for various initial populations but still we may not get optimal solution.

• Another major shortcoming of our meta-heuristic approach is the variance with respect to the random seed and hyper parameters. The optimal solution only arrives for certain seeds in a fixed number of iterations. Certain seeds may entirely fail to provide a feasible solution

Thank You