Binary task assignment in CPLEX

Roberto Meroni* Hugo Sanz González†

October 2023

Consider the problem of distributing the workload required to perform a set of tasks T among a set of computers C. Each task $t \in T$ requires $r_t \geq 0$ compute units, and each computer $c \in C$ has $r_c \geq 0$ compute units available. Let P_1 be a linear program where portions of a task may be assigned to different computers, and let P_2 be a mixed integer linear program where a task is assigned to exactly one computer.

1. *Implement the P_2 model in OPL and solve it using CPLEX.*

The difference between the models P_1 and P_2 is in the definition of the decision variable x_{tc}, for $t \in T$ and $c \in C$. In the first case, $x_{tc} \in [0, 1]$ represents the fraction of work required to complete task t that is assigned to computer c. Note that in P_2 a task is assigned completely to a single computer, hence $x_{tc} \in \{0, 1\}$ is a binary decision variable instead. The corresponding definition in OPL reads

```
dvar boolean x_tc[t in T, c in C];
```

The existing set of constraints $\sum_{c \in C} x_{tc} = 1$, for $t \in T$, ensures that a feasible solution processes all tasks.

Suppose that the work required is $r_{t_1} = 261.27$, $r_{t_2} = 560.89$, $r_{c_3} = 310.51$, $r_{c_4} = 105.80$; and the compute availability is $r_{c_1} = 505.67$, $r_{c_2} = 503.68$, $r_{c_3} = 701.78$. CPLEX then finds the optimal solution $x_{t_1c_1} = x_{t_2c_3} = x_{t_3c_2} = x_{t_4c_1} = 1$ and $x_{tc} = 0$ for all other task-computer pairs, in which the load of the highest-loaded computer is roughly 79.9%.

2. *Compare P_1 and P_2 in terms of the value of the optimal solution, solving algorithm, solving time, and number of variables and constraints.*

Table 1 lists these statistics for the two models. As remarked before, the only difference between P_1 and P_2 is the domain of the decision variables: whereas $x_{tc} \in [0, 1]$ in the former, we have $x_{tc} \in \{0, 1\}$ in the latter for every $t \in T$ and $c \in C$. Thus, every feasible solution to P_2 lies in the feasible region of P_1. It follows that the value of the objective function (to be minimized) at an optimal solution of P_1 is less than or equal to that of any optimal solution to P_2. Indeed we have $0.724 \leq 0.799$.

The Simplex algorithm is not directly applicable to linear programs that contain integer variables; the CPLEX environment uses the more costly *Branch & cut* technique to solve P_2. The number of variables and constraints in the

*roberto.meroni@estudiantat.upc.edu
†hugo.sanz@estudiantat.upc.edu

Model	Objective	Algorithm	Runtime (ms)	Variables	Constraints
P_1	0.724	Simplex	0.8	13	10
P_2	0.799	B&C	4.3	13	10

Table 1: Certain statistics of the task assignment models.

two problems is the same, the only difference being that in P_1 all variables are continuous whereas in P_2 only 1 out of 13 is continuous.

3. *Solve P_2 with the new data file, where a new task has been added. Analyze the obtained results and compare them when solving P_1 with the same data.*

The number of compute units required by the second task $r_{t_2} = 560.89$ is greater than the compute availability of the first and the second computer, so this task must be assigned to the third computer. The remaining availability of this computer may only be taken up in part by the fourth task, with $r_{t_4} = 105.8$. Assigning t_4 to any other computer is a less optimal choice (for obtaining a feasible solution) and the total compute units required by any pair of the remaining three tasks t_1, t_3, t_5 exceeds the capacities of both computer c_1 and c_2. Therefore CPLEX shows that the problem P_2 is infeasible.

In P_1 the load of each task can be distributed among different computers, and CPLEX finds an optimal solution with objective value 0.92522.

4. *Modify the P_2 model to allow rejecting tasks; that is, some tasks might not be processed.*

A task need no longer be assigned to a computer, hence the first set of constraints $\sum_{c\in C} x_{tc} = 1$ becomes $\sum_{c\in C} x_{tc} \leq 1$ for all $t \in T$. In the OPL language,

```
forall(t in T)
    sum(c in C) x_tc[t,c] <= 1;
```

Let $n = |T|$ be the total number of tasks, and observe that the number of assigned tasks is $\sum_{t\in T, c\in C} x_{tc}$. The number of rejected tasks is thus $n - \sum_{t\in T, c\in C} x_{tc}$, which we may force to be less than or equal to a threshold $k \geq 0$ through the OPL constraint

```
nTasks - sum(t in T) sum(c in C) x_tc[t, c] <= K;
```

Table 2 lists the optimal solutions given by CPLEX when up to $0 \leq k \leq 5$ tasks can be rejected. Assigning a task to a computer may increase the load of the highest loaded computer, so rejecting a task if permissible is usually a good choice. In the limit as k approaches n, the optimal solution is to assign no task.

5. *Modify the objective function so as to minimize the amount of non-served load.*

The total amount of load ready to be assigned is $\sum_{t\in T} r_t$, out of which only $\sum_{t\in T, c\in C} r_t x_{tc}$ is served. Hence the amount of non-served load—the objective to be minimized—equals

$$\sum_{t\in T} r_t - \sum_{t\in T}\sum_{c\in C} r_t x_{tc}.$$

k	Unassigned	Objective	Load (%)		
			c_1	c_2	c_3
0		-			
1	t_2	0.642	61.4	51.9	64.2
2	t_2, t_5	0.517	51.7	21.0	44.3
3	t_2, t_3, t_5	0.372	20.9	0	37.2
4	t_1, t_2, t_3, t_5	0.151	0	0	15.1
5	$t_1, \ldots, t_5$	0	0	0	0

Table 2: Optimal solutions to a variant of P_2 where up to k tasks may be unassigned, using the new data file.

k	Unassigned	Served	Load (%)		
			c_1	c_2	c_3
0		-			
1	t_1	1321.9	89.1	61.6	79.9
2	t_1	1321.9	68.2	61.6	95.0
3	t_1	1321.9	68.2	61.6	95.0
4	t_1	1321.9	89.1	61.6	79.9
5	t_1	1321.9	89.1	61.6	79.9

Table 3: Optimal solutions to a variant of P_2 where the objective is to maximize the amount of load served and up to k tasks may be unassigned, using the new data file.

The first term is constant, so the problem of minimizing the previous quantity is equivalent to the MIP where $-\sum_{t \in T} \sum_{c \in C} r_t x_{tc}$ is minimized. This in turn is equivalent to the problem of maximizing $\sum_{t \in T} \sum_{c \in C} r_t x_{tc}$: minimizing the amount of non-served load is the same as maximizing the amount of served load. The corresponding OPL statement is thus

```
maximize sum(c in C) sum(t in T) x_tc[t, c] * rt[t];
```

Table 3 shows the solutions to the task assignment problem in Exercise 1 given by CPLEX under this new formulation, for $0 \leq k \leq 5$. Now it is always advantageous to assign a task to a computer, so once the feasible region is nonempty (which here requires $k \geq 1$) the number of unassigned tasks remains constant as k increases. The objective value is therefore the same for all listed solutions, but it is interesting to note that for $k \in \{1, 4, 5\}$ the Branch & cut procedure finds an optimal solution in which t_4 is assigned to c_1, whereas for $k \in \{2, 3\}$ it gives $x_{t_4 c_3} = 1$ instead (this is due to the fact that there are multiple optimal solutions and the algorithm stops at the first optimal solution found, so it depends only on which branch is taken. This choice affects the load distribution across the computers, which the objective from the previous exercise managed to even out. Combining the objectives from the two exercises is probably a better choice for both reducing the amount of unassigned work and balancing the workload assigned to computers.

6. *Compare all three models in terms of number of variables, constraints and*

	Variables				
Ex.	Binary	Continuous	Constraints	k	Runtime (ms)
3	15	1	11		3.6
4	15	1	12	0	14.5
				2	73.4
				4	19.4
5	15	0	9	0	12.2
				2	97.8
				4	99.7

Table 4: Comparison of the variants of P_2 in Exercises 3 to 5.

execution time.

Table 4 shows the values of these statistics for the three variants of P_2 studied in Exercises 3 to 5. Note that the three models have the same number of binary variables, but those that minimize the load of the highest loaded computer need to introduce the real variable z and bound it through $|C|$ constraints; only the last model is a pure integer program. The last two models place an upper bound on the number of unassigned tasks through an additional constraint.

As remarked before, CPLEX employs the Branch & cut search-based algorithm for finding an optimal solution to a mixed integer linear program. The execution time of this procedure depends on the shape of the feasible region, the number of cuts found, and the number of solved subproblems; this fact explains the variability of the running time when solving the models with similar input data. But it is clear that lifting the "all tasks must be assigned" restriction (that is, setting $k > 0$) increases the search space and worsens the running time by almost an order of magnitude.