# On the Cooling-Aware Workload Placement Problem

**Paolo Cremonesi, Andrea Sansottera**
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Via Ponzio 34/5
20133 Milano, Italy
paolo.cremonesi@polimi.it, sansottera@elet.polimi.it

**Stefano Gualandi**
Dipartimento di Matematica
Università degli Studi di Pavia
via Ferrata, 1, 27100 Pavia, Italy
stefano.gualandi@unipv.it

**Abstract**

This paper proposes a new challenging optimization problem, called COOLING-AWARE WORKLOAD PLACEMENT PROBLEM, that looks for a workload placement that optimizes the overall data center power consumption given by the sum of the server power consumption and of the computer room air conditioner power consumption. We formulate CWPP as a Mixed Integer Non Linear Problem using a cross-interference matrix that links the workload placement to the cold air temperature. Since state-of-the-art Mixed Integer Non Linear solvers can solve to optimality only the smallest instances, we devised two heuristics to obtain good feasible solutions: (i) a heuristic algorithm based on an integer linear relaxation of the problem, and (ii) a Variable Neighborhood Search algorithm. Both heuristic algorithms are evaluated against the best lower bounds obtained with a Mixed Integer Non Linear solver. Preliminary computational results show that both heuristics provide solutions that have a small percentage gap from the optimal solutions.

## 1 Introduction

In a heterogeneous data center, power-aware workload placement, that is the problem of assigning workloads to servers, can reduce power consumption by prioritizing the most efficient servers. Unfortunately, this strategy can lead to *hot spots*, which require to lower the temperature of the the Computer Room Air Conditioner (CRAC) in order to meet the thermal specifications of the servers, hence increasing the power spent for cooling (Banerjee et al. 2010).

This paper proposes a new challenging optimization problem, herein called COOLING-AWARE WORKLOAD PLACEMENT PROBLEM (CWPP), that looks for a workload placement that optimizes the overall data center power consumption given by both the power consumption of the servers and of the CRAC. That is, in addition to the power server consumption, we consider the heat distribution in the server room and the efficiency of the CRAC at different temperatures of the air supplied. As in previous work, we characterize the workloads to be allocated by their performance characteristics (i.e., service demands and arrival rates) and include constraints on the maximum tolerated mean response times, as defined in service level agreements.

In order to obtain the heat distribution for a given server power distribution, an expensive Computational Fluid Dynamic (CFD) simulation can be performed. This approach, however, is clearly unfit to finding the optimal workload placement, since a CFD simulation would be required to evaluate the objective function. In order to make the problem tractable, we adopt the linear heat flow model proposed in (Tang et al. 2006). The general idea of this approach is to compute a *cross-interference matrix*, which characterizes the heat exchanged between each pair of servers through a small number of CFD simulations. This matrix can be used for the fast evaluation of the temperatures at the server inlets for a given workload placement and cold air temperature.

We formulate CWPP as a Mixed Integer Non Linear Problem (MINLP), with a non convex objective function and linear constraints over both integer and continuous variables. The cross-interference matrix is used to define a set of linear constraints that links the workload placement to the cold air temperature. Since state-of-the-art MINLP solvers can solve to optimality only the smallest instances, we devised two heuristics to obtain good feasible solutions: (i) a heuristic algorithm based on an integer linear relaxation of the problem, and (ii) a Variable Neighborhood Search (VNS) algorithm. Both heuristic algorithms are evaluated against the best lower bounds obtained by solving CWPP with a MINLP solver. The solution obtained with the two heuristics are within a small percentage gap from the optimal solutions.

In literature, many works have dealt with consolidation, i.e. the process of combining the workloads of several different servers on a set of target servers. Rolia et al. (Gmach et al. 2009) suggest an integer program for allocation problems in a data center. Linear and non-linear integer programming models for consolidation are presented in (Anselmi, Amaldi, and Cremonesi 2008). These models aim at minimizing the number of servers used while satisfying constraints on end-to-end response times. A similar minimization problem is formulated in (Dhyani, Gualandi, and Cremonesi 2010), where response time constraints are dropped, but rules related to availability and compatibility requirements are introduced. In (Moore et al. 2005), the authors tackle thermal-aware placement compute-intensive batch jobs, characterizing heat recirculation among servers. This idea is further developed in (Tang et al. 2006), where the cross-interference

approach is proposed. Most of the work exploiting heat recirculation models for cooling-aware workload placement do not consider servers with heterogeneous power and performance characteristics (Tang, Gupta, and Varsamopoulos 2007; Pakbaznia and Pedram 2009). Hence, they minimize heat recirculation or maximize the temperature of the cold air and do not deal with the trade-off between server power consumption and cooling power consumption, which introduces the non convexity in our mixed integer formulation. Moreover, performance requirements are often expressed in very simple terms, e.g. the number of servers needed to operate at certain frequency (Pakbaznia and Pedram 2009) or the number of CPU cores needed (Moore et al. 2005). On the other hand, we integrate capacity and response time constraints within our model, based on the arrival rates and service demands of the different workloads.

The outline of the paper is as follows. Section 2 describes the workload placement problem. Section 3 presents the MINLP formulation and present a simple Integer Linear Programming relaxation. Section 4 introduces the VNS algorithm and the greedy algorithm used to find a feasible solution. Finally, Section 5 presents preliminary computational results on a set of realistic instances, and Section 6 concludes the paper.

## 2 Problem statement

In the following, we describe the three main aspects that are considered in our optimization problem: data center power consumption, heat recirculation and the performance model of the servers and workloads.

### Data center power consumption

The power consumption of a data center can be seen as the sum of two main components, the power consumed by the servers and by the CRAC. For the sake of simplicity and without loss of generality, we do not consider the power consumption of other devices, such as the power distribution units.

The power drawn by a set of servers $S$ is given by the sum of the power consumption $p_s$ of each server $s \in S$, that is, $P_{\mathrm{S}} = \sum_{s \in S} p_s$. Similarly to Mantis (Economou, Rivoire, and Kozyrakis 2006), we adopt a power model based on resource utilization and consider the most power hungry component of the computer system, the CPU:

$$p_s = P_{idle,s} + P_{busy,s} u_s \,, \tag{1}$$

where $u_s \in [0, 1]$ is the CPU utilization. Least squares regression can be applied to the data from standard power benchmarks (e.g., SPECpower[1]) to estimate the two parameters $P_{idle,s}$ and $P_{busy,s}$.

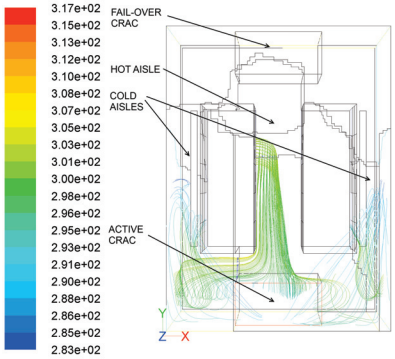
The amount of heat the CRAC removes from the computer room is equal to $P_{\mathrm{S}}$, since we assume that steady state conditions hold, and all the power drawn by the servers is dissipated as heat. The ratio between the heat removed and the power drawn by a heat pump is called the Coefficient of Performance (COP). This coefficient depends directly on the temperature of the cold air supplied by the CRAC. We adopt the COP curve estimated for the CRAC units at the HP Labs Utility Data Center (Moore et al. 2005) and we write the power consumption of the CRAC as follows:

$$P_{\mathrm{CRAC}} = \frac{P_{\mathrm{S}}}{b_1 z^2 + b_2 z + b_3} \,, \tag{2}$$

with $z = T_{sup} - 273.15$ K, where $T_{sup}$ is the temperature of the air supplied by the CRAC and the three coefficients $b_1$, $b_2$, $b_3$ minimize the mean square error of the COP curve (i.e., they carry no physical meaning). The coefficients in equation (2) are such that $P_{\mathrm{CRAC}}$ monotonically decreases with $T_{sup}$.

[1] http://www.spec.org/power_ssj2008/

Figure 1: This figure, obtained by a CFD simulation, shows how the air exiting a server recirculates into the inlets of the other servers.




### Heat recirculation

Data center layouts are designed with alternating hot and cold aisles, with the goal of maximizing the amount of cold air drawn by the server and the hot air exhaust drawn by the CRAC units. However, some of the air inevitably recirculates among servers, as shown in Figure 1. The figure, obtained using a CFD simulation, shows the air flow exiting one of the server. The layout comprise two rows of racks, separated by a hot isle, and two CRAC units. Only one of the CRAC units is active and draws a significant amount of air exiting from the server. However, some of the hot air exhaust recirculates to the intake of the other servers, hence impacting the inlet temperatures. The abstract heat flow model, presented in (Tang et al. 2006), enables the characterization of heat recirculation among servers for a given data center layout.

For a given server $s$, let $T_{in,s}$ be the inlet temperature and $T_{out,s}$ the outlet temperature of the air. According to the abstract heat flow model, given a set of servers $S$, the temperature $T_{in,i}$ at the inlet of a server $i \in S$ can be written as a weighted sum of the temperatures at the servers outlets $T_{out,j}$ for all $j \in S$ and the temperature of the cold air supplied by the CRAC $T_{sup}$:

$$T_{in,i} = \sum_{j \in S} A_{ji} T_{out,j} + (1 - \sum_{j \in S} A_{ji}) T_{sup} \tag{3}$$

where the coefficients $A_{ji}$ represent the fraction of heat exhausted by server $j$ that recirculates to the inlet of server

$i$. Although, the inlet temperatures of the servers $T_{in,s}$ must not exceed a critical threshold $T_{crit,s}$, specified by the server manufacturer, that is $T_{in,s} \leq T_{crit,s}$.

Assuming that all the power drawn by a server is dissipated as heat and that the thermal exchange between a server and the computer room happens mostly through convection, due to the law of energy conservation,

$$p_s = \phi_s \rho C_p (T_{out,s} - T_{in,s}) \,, \tag{4}$$

where $\rho$ and $C_p$ are the density and specific heat of the air, while $\phi_s$ is the volumetric air flow rate of the server. For simplicity of notation we let $K_s = \phi_s \rho C_p$. Hence, we can rewrite (3) as

$$T_{in,i} = \sum_{j \in S} A_{ji} \left( T_{in,j} + \frac{p_j}{K_j} \right) + (1 - \sum_{j \in S} A_{ji}) T_{sup} \,. \tag{5}$$

### Performance model

Let $C$ be the set of workloads that must be allocated and let $S$ be the set of available servers. We adopt an open queueing model and for each workload $c$, the arrival rate is denoted by $\lambda_c$. For each workload and server pair, we know the service demand $d_{cs}$ on a single CPU core. The number of CPU cores installed on each server is denoted by $n_s$.

According to the utilization law (Lazowska et al. 1984), for each server $s \in S$, the CPU utilization is defined as

$$u_s = \sum_{c \in C_s} \frac{\lambda_c d_{cs}}{n_s} \,, \tag{6}$$

where $C_s \subseteq C$ is the set of workloads allocated on $s$. In order to not exceed server capacity, we require that $u_s \leq 1$.

To estimate the response time $r_c$ for a workload $c$ allocated on $s$, we use the approximation for multi-server queues with exponentially distributed service times and inter-arrival times described in (Menasce and Almeida 2001):

$$r_c = d_{cs} \frac{n_s - 1}{n_s} + \frac{d_{cs}/n_s}{1 - u_s} \,. \tag{7}$$

In order to meet service level agreements, we require that $r_c \leq R_c^{(max)}$, where $R_c^{(max)}$ is the maximum tolerated mean response time.

## 3 Mathematical Programming Formulation

We present in this section a Mixed Integer Nonlinear Programming formulation of the COOLING-AWARE WORKLOAD PLACEMENT PROBLEM. The MINLP formulation is used to compute exact solutions of small instances and to obtain lower bounds of the bigger instances. In the following paragraph we review the notation used to stated the MINLP model.

CWPP is the problem of assigning each workload to exactly one of the server available, in such a way to satisfy the thermal constraints of the servers and the computational requirements of each workload, while minimizing the overall power consumption. The power consumption is a non linear function of the overall power consumed by the servers and the power consumed by the CRAC, as given in equations (1) and (2).

Let $x_{cs}$ be a 0–1 decision variable equals to 1 whenever the workload $c$ is assigned to the server $s$. Let $y_s$ be a positive variable representing the temperature at the inlet of the server $s$ and let $z$ be the variable representing the temperature given by the air conditioning system.

For the sake of simplicity, we define two auxiliary variables that depends only on the assignment variables $x_{cs}$: the utilization variable $u_s$ and the server power variable $p_s$. These two auxiliary variables are defined as follows:

$$u_s = \sum_{c \in C} \frac{\lambda_c \, d_{cs}}{n_s} \, x_{cs} \tag{8}$$

$$p_s = P_{idle,s} + P_{busy,s} \, u_s \tag{9}$$

In addition, we define the following parameters. Let $B_{cs} = \frac{n_s R_c^{(max)}}{d_{cs}} - n_s + 1$ and let $b_1$, $b_2$, and $b_3$ three coefficients that depend on the CRAC. The temperature exiting from the CRAC is bounded by $z^{lb}$ and $z^{ub}$.

Using this notation, the CWPP problem is formulated as the following MINLP:

$$\min \left( 1 + \frac{1}{b_1 z^2 + b_2 z + b_3} \right) \sum_{s \in S} p_s \tag{10}$$

$$\text{s.t.} \sum_{s \in S} x_{cs} = 1, \qquad \forall c \in C, \tag{11}$$

$$\frac{1}{B_{cs}} x_{cs} + u_s \leq 1, \qquad \forall c \in C, \forall s \in S, \tag{12}$$

$$z = \frac{y_s - \sum_{s' \in S} A_{ss'} (y_{s'} + \frac{p_{s'}}{K_{s'}})}{1 - \sum_{s' \in S} A_{ss'}}, \quad \forall s \in S, \tag{13}$$

$$x_{cs} \in \{0, 1\}, \qquad \forall c \in C, \forall s \in S, \tag{14}$$

$$0 \leq y_s \leq T_{crit,s}, \qquad \forall s \in S, \tag{15}$$

$$z^{lb} \leq z \leq z^{ub}. \tag{16}$$

The objective function is (10) is clearly nonlinear and non convex; the auxiliary variables $p_s$ depends only on the assignment variables $x_{cs}$, as defined in (9). Constraints (11) are the assignment constraints and guarantee that each workload is assigned to a single server. Constraints (12) are the response time constraints that force each workload to be assigned to a server that can execute the workload within its maximum response time. Note that the term $u_s$ depends only by the assignment variables $x_{cs}$. Constraints (13) couple the air conditioning (controlled) temperature $z$ to the temperatures $y_s$ of each server, and the workload assignment via the auxiliary variable $p_{s'}$.

A naive linear lower bound to problem (10)–(16) is obtained by considering only the linear term in the objective function (10), that is to consider only the power consumption given by the servers:

$$\min \sum_{s \in S} p_s \tag{17}$$

This lower bound is in general very weak, and the optimality gap can be very big.

However, it is useful to solve the problem of finding the highest possible value of $z$ such that a feasible workload

placement exists. In order to find such value of temperature, it is sufficient to solve the following Integer Linear Programming problem:

$$z^* = \max\ z, \quad \text{s.t. (11)–(16)} \tag{18}$$

The optimal value $z^*$ can be used in at least two ways.

First, if we fix the value of $z$ to $z^*$, then problem (10)–(16) becomes an Integer Linear Program, and its solution will give an upper bound.

Second, the upper bound on $z$ can be restricted to $z^*$ by setting $z^{ub} = z^*$. Such restriction is in practice very important in order to obtain a tight lower bound using a MINLP solver such Couenne, that is based on bound tighten techniques (Belotti et al. 2009).

## 4 Variable Neighborhood Search

In this section we introduce a Variable Neighborhood Search (VNS) algorithm for CWPP. The VNS algorithm is based on the observation that given any workload placement $\mathbf{X} \in \{0,1\}^{|C|\times|S|}$, the server utilization $u_s$, the server power consumption $p_s$, and the optimal cold air temperature $z$ can be computed as described in Section 2. Hence, the feasible solution space explored by the heuristics is $\mathcal{Q} \subseteq \{0,1\}^{|C|\times|S|}$.

The initial feasible solution is computed with a greedy algorithm. First, the workloads are sorted according to the ratio between the maximum allowed response time and the standardized service demand. A low value of this ratio indicates that the workload requires more headroom on the server in order to meet its performance requirements and hence we try to allocate it early. Then, the algorithm selects a workload $c$ at a time, and assigns $c$ to the server which yields the least increase in the objective function without violating constraints (12).

In order to improve the initial feasible solution, we apply a local search procedure that explores with a best improvement strategy the neighborhood obtained by considering all the possible moves of a workload $c$ to any another server $s$. In practice, we select the best move by exploring all solutions $\mathbf{X}' \in \mathcal{Q}$ such that

$$|\{c \in C : x_{c,\cdot} \neq x'_{c,\cdot}\}| = 1\ . \tag{19}$$

The pseudo-code of our basic Local Search procedure is shown in Algorithm 1.

Once the Local Search procedure is trapped in a local optimum, that is, no improvement move exists in the neighborhood of $\mathbf{X}$ given by (19), we consider a different neighborhood, following the basic idea of the Variable Neighborhood Search approach. We consider a series of neighborhood structures $\mathcal{N}_k(\mathbf{X})$ for $k \in \{1,\ldots,k_{max}\}$, with $k_{max} = \left\lfloor \frac{|C|}{10} \right\rfloor$. The neighborhood $\mathcal{N}_k(\mathbf{X})$ is defined as the set of solutions $\mathbf{X}' \in \mathcal{Q}$ such that

$$|\{c \in C : x_{c,\cdot} \neq x'_{c,\cdot}\}| \leq k\ . \tag{20}$$

In practice, we select randomly $k$ workloads and we try to randomly relocated these workloads to different servers. Our VNS randomly explore the neighborhood $\mathcal{N}_k(\mathbf{X})$ until either a new feasible assignment is found or until a time limit is expired. In the first case, we continue we the basic local search procedure by exploring the neighborhood (19) with a best improvement strategy. In the latter case, i.e. the time out is expired, we increase the value of $k$. Algorithm 2 shows the pseudo-code of our VNS algorithm.

**Algorithm 1** The local search procedure

```
procedure LocalSearch
    currentObj ← computeObjective()
    continue ← true
    while continue do
        improvement ← false
        bestObj ← currentObj
        for c ∈ C do
            for s ∈ S do
                if x_cs = 0 and x_cs = 1 is feasible then
                    s̃ ← j ∈ S : x_cj = 1
                    x_cj ← 0 for all j ∈ S, j ≠ s
                    x_cs ← 1
                    obj ← computeObjective()
                    if obj < bestObj then
                        improvement ← true
                        bestObj ← obj
                        c* ← c
                        s* ← s
                    end if
                    x_cs ← 0
                    x_cs̃ ← 1
                end if
            end for
        end for
        if improvement then
            x_c*s* = 1
            x_c*j = 0 for all j ∈ S, j ≠ s*
        else
            continue ← false
        end if
    end while
end procedure
```

## 5 Computational Results

The MINLP model and the VNS algorithm were tested on a set of realistic random instances. All the computational tests were executed on the same computer, characterized by a Core i7 920 CPU and 12 GB of DDR3 memory.

Test cases with either 10 servers or 40 servers were considered. Hence, two different cross-interference matrices were obtained, spending thousands of CPU hours for the CFD simulations. Test cases are further differentiated by the number of workloads (2, 4 or 10 times the number of servers) and the overall data center utilization (0.3, 0.5 or 0.7). For each test case, 10 different instances of the optimization problem were generated and tested. The other parameters vary across different problem instances. In particular, the number of CPUs, the idle power, and the busy power were generated according to uniform discrete and continuous distributions. Let $\mathcal{U}_d\{h_1,\ldots,h_n\}$ denote the uniform

**Algorithm 2** The Variable Neighborhood Search heuristic

```
procedure VNS
    x*_cs ← x_cs for all (c, s) ∈ C × S
    bestObj ← computeObjective()
    k ← 0
    while below time limit do
        LocalSearch()
        obj ← computeObjective()
        if obj < bestObj then
            x*_cs ← x_cs for all (c, s) ∈ C × S
            bestObj ← obj
            k ← 0
        end if
        x_cs ← x*_cs for all (c, s) ∈ C × S
        k ← min(k + 1, k_max)
        successful ← false
        while below time limit and not successful do
            successful ← perturbate(k)
            if not successful then
                x_cs ← x*_cs for all (c, s) ∈ C × S
            end if
        end while
    end while
end procedure
```

discrete distribution over the set of values $h_1, \ldots, h_n$ and $\mathcal{U}(a, b)$ denote the uniform continuous distribution over the interval $[a, b]$. The parameters of the distributions were chosen as follows:

$$n_s \sim \mathcal{U}_d\{60, 80, 120, 128, 160, 192\} \quad \forall s \in S$$
$$P_{idle,s} \sim \mathcal{U}(200, 400)\mathrm{W} \quad \forall s \in S$$
$$P_{busy,s} \sim \mathcal{U}(15, 25) \cdot n_s\mathrm{W} \quad \forall s \in S\,.$$

In order to generate the service demands, we considered a reference server $\tilde{s}$ with speedup factor $f_{\tilde{s}} = 1$ and one CPU core, i.e. $n_{\tilde{s}} = 1$. We generated the service demands of the workloads on this reference server as follows:

$$d_{c,\tilde{s}} \sim \mathcal{U}(0.05, 0.1) \quad \forall c \in C\,.$$

In order to appropriately scale the service demands on the different servers and to obtain the service demands for every workload-server pair, we define the speedup factor $f_s$ of a server $s$ with respect to the reference server $\tilde{s}$ as

$$f_s = \frac{d_{c,s}}{d_{c,\tilde{s}}} n_s\,,$$

which we assume to be constant for all workloads $c \in C$. The speedup factors were generated according to a uniform distribution

$$f_s \sim \mathcal{U}(0.8, 1.6) \cdot n_s \quad \forall s \in S.$$

The constraints on the mean response times were set to

$$R_c^{(max)} = 0.15 \quad \forall c \in C\,.$$

If $U^{(total)}$ represents the overall utilization of the data center, the total standard CPU capacity consumed is $G^{(total)} = U^{(total)} \sum_{s \in S} f_s$. This capacity was partitioned across workloads according to randomly generated weights $g_c \sim \mathcal{U}(0, 1)$, setting the arrival rates as follows:

$$\lambda_c = \frac{\frac{g_c}{\sum_{j \in C} g_j} G^{(total)}}{d_{c,\tilde{s}}} \quad \forall c \in C\,.$$

Table 1: Bounds and VNS results for the 10 random instances with 10 servers, 20 applications and 0.3 data center utilization.

| Instance | Upper | Lower | VNS | Gap |
|---|---|---|---|---|
| 1 | 10553.9 | 10551.8 | 10553.9 | 0.02% |
| **2** | **9445.1** | **9445.1** | **9445.6** | **0.00%** |
| 3 | 9298.0 | 9298.0 | 9298.8 | 0.01% |
| 4 | 9134.1 | 9133.5 | 9135.9 | 0.03% |
| **5** | **8964.5** | **8964.5** | **8965.1** | **0.01%** |
| **6** | **9044.9** | **9044.9** | **9045.1** | **0.00%** |
| 7 | 9661.3 | 9657.9 | 9662.2 | 0.04% |
| 8 | 10922.6 | 10917.1 | 10922.5 | 0.05% |
| **9** | **8272.1** | **8272.1** | **8272.3** | **0.00%** |
| 10 | 8884.7 | 8883.3 | 8893.1 | 0.11% |

## Optimal Solutions and Lower Bounds

The first set of experiments used the MINLP model (10)–(16) to compute optimal solutions of the smaller instances and lower bounds on the bigger instances. In order to assess the quality of the heuristics, we compute the gap of the solution from the optimal solution when it is available; otherwise we compute the gap from the best known lower bound. Therefore, the gaps reported in Tables 1,2,3,4 are estimated (pessimistic) gaps.

All the problem instances were solved using the Couenne MINLP solver (Belotti et al. 2009) with a time limit of 600 seconds. Before solving the MINLP model, we solved problem (18) in order to get the tightest upper bound on $z$. Problem (18) was solved using CPLEX with a timeout of 100 seconds; if the timeout was expired, we used the best upper bound on $z$ found at the time limit to set the value of $z^{ub}$. Adopting this strategy, Couenne was able to find or get very close to the optimal solution value in the least challenging test case, as shown in Table 1. Instances in which the optimum was found are marked in bold.

## MILP-based Heuristic

A simple method to compute upper bounds consists in first to optimize over $z$ solving the ILP (18), then to bound $z$ between $z^* - \epsilon$ and $z^*$, and then to solve the ILP given by the objective function (17) along with the constraints (11)–(16). A second strategy consists in iteratively enlarging the bound on $z$, by considering, for instance, $z^* - 2\epsilon$, $z^* - 3\epsilon$, and so on. In practice, we used five successive intervals with $\epsilon = 0.15$. Despite being simple, these two simple strategies yield good upper bounds, as shown in Table 2. The execution time, however, is 120 seconds, twelve times larger than the time limit set for the VNS procedure and results quickly get worse as the number of servers is increased.

Table 2: Mean results of the MILP heuristics (10 servers).

| $\|C\|$ | $U^{(total)}$ | MILP-1 | Gap 1 | MILP-2 | Gap 2 |
|---|---|---|---|---|---|
| 20 | 0.3 | 9968.3 | 5.97% | 9462.0 | 0.51% |
| 20 | 0.5 | 13809.0 | 5.03% | 13313.9 | 1.18% |
| 20 | 0.7 | 20677.5 | 1.28% | 20464.5 | 0.24% |
| 40 | 0.3 | 9890.4 | 6.42% | 9325.2 | 0.21% |
| 40 | 0.5 | 15189.3 | 5.55% | 14527.1 | 0.87% |
| 40 | 0.7 | 19447.8 | 1.00% | 19289.5 | 0.22% |
| 100 | 0.3 | 10161.7 | 5.77% | 9716.6 | 1.13% |
| 100 | 0.5 | 14811.6 | 6.13% | 14267.4 | 2.20% |
| 100 | 0.7 | 19145.8 | 3.02% | 18834.0 | 1.34% |

Table 3: Mean results of the VNS heuristic (10 servers).

| $\|C\|$ | $U^{(total)}$ | VNS | Avg Gap | Min Gap | Max Gap |
|---|---|---|---|---|---|
| 20 | 0.3 | 9419.4 | 0.03% | 0.00% | 0.11% |
| 20 | 0.5 | 13170.3 | 0.14% | 0.03% | 0.37% |
| 20 | 0.7 | 20519.8 | 0.51% | 0.15% | 0.94% |
| 40 | 0.3 | 9320.7 | 0.15% | 0.01% | 1.22% |
| 40 | 0.5 | 14498.8 | 0.66% | 0.03% | 3.68% |
| 40 | 0.7 | 19269.9 | 0.12% | 0.07% | 0.14% |
| 100 | 0.3 | 9671.7 | 0.71% | 0.01% | 1.74% |
| 100 | 0.5 | 14201.8 | 1.77% | 0.19% | 3.69% |
| 100 | 0.7 | 18811.3 | 1.22% | 0.09% | 2.87% |

### VNS

The VNS heuristic was implemented in C++ and run with a time limit of 10 seconds. The results obtained on the test cases with 10 servers are summarized in Table 3. Using the lower bound provided by the MINLP approach, we show that the solutions found by VNS are, on average, within 1.77% of the optimum. Maximum gaps never exceed 3.69% across all the 90 problem instances.

The VNS heuristic was also run on the more challenging test cases, in which 40 servers and up to 400 workloads are present. Results are shown in Table 4. These instances turn out to be very challenging, even for getting a lower bound. As part of the future work, we plan to improve the MINLP approach in order to get lower bounds also on the 40 servers instances.

## 6 Conclusions

In this paper, we presented a non-linear mixed integer program which aims at minimizing the overall power consumption of a data center. Constraints related to thermal specifications of the servers, capacity and response time requirements were considered. To tackle the problem, we proposed two MILP-based heuristics and a VNS heuristic. A strategy to effectively compute good lower bounds was also devised. Extensive computational results show that the VNS heuristic achieves average optimality gaps below $2\%$.

Table 4: Mean results of the VNS heuristic (40 servers).

| $\|C\|$ | $U^{(total)}$ | VNS | Gap |
|---|---|---|---|
| 80 | 0.3 | 41076.5 | 2.33% |
| 80 | 0.5 | 63540.1 | - |
| 80 | 0.7 | 98771.8 | - |
| 160 | 0.3 | 40895.2 | - |
| 160 | 0.5 | 64510.3 | - |
| 160 | 0.7 | 101918.3 | - |
| 400 | 0.3 | 40483.1 | - |
| 400 | 0.5 | 64530.7 | - |
| 400 | 0.7 | 97556.1 | - |

## References


Anselmi, J.; Amaldi, E.; and Cremonesi, P. 2008. Service consolidation with end-to-end response time constraints. In *Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference*, 345 –352.

Banerjee, A.; Mukherjee, T.; Varsamopoulos, G.; and Gupta, S. 2010. Cooling-aware and thermal-aware workload placement for green hpc data centers. In *Green Computing Conference, 2010 International*, 245 –256.

Belotti, P.; Lee, J.; Liberti, L.; Margot, F.; and Wachter, A. 2009. Branching and bounds tightening techniques for non-convex minlp. *Optimization Methods Software* 24:597–634.

Dhyani, K.; Gualandi, S.; and Cremonesi, P. 2010. A constraint programming approach for the service consolidation problem. In *Proc CPAIOR*, volume 6140 of *LNCS*. Springer Berlin / Heidelberg. 97–101.

Economou, D.; Rivoire, S.; and Kozyrakis, C. 2006. Full-system power analysis and modeling for server environments. In *In Workshop on Modeling Benchmarking and Simulation (MOBS*.

Gmach, D.; Rolia, J.; Cherkasova, L.; and Kemper, A. 2009. Resource pool management: Reactive versus proactive or let's be friends. *Comput. Netw.* 53:2905–2922.

Lazowska, E. D.; Zahorjan, J.; Graham, G. S.; and Sevcik, K. C. 1984. *Quantitative system performance: computer system analysis using queueing network models*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Menasce, D. A., and Almeida, V. 2001. *Capacity Planning for Web Services: metrics, models, and methods*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st edition.

Moore, J.; Chase, J.; Ranganathan, P.; and Sharma, R. 2005. Making scheduling "cool": temperature-aware workload placement in data centers. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, 5–5. Berkeley, CA, USA: USENIX Association.

Pakbaznia, E., and Pedram, M. 2009. Minimizing data center cooling and server power costs. In *Proceedings of the 14th ACM/IEEE international symp. on Low power electronics and design*, ISLPED '09, 145–150. New York, NY, USA: ACM.

Tang, Q.; Mukherjee, T.; Gupta, S.; and Cayton, P. 2006. Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters. In *Intelligent Sensing and Information Processing, 2006. ICISIP 2006. Fourth International Conference on*, 203 –208.

Tang, Q.; Gupta, S.; and Varsamopoulos, G. 2007. Thermal-aware task scheduling for data centers through minimizing heat recirculation. In *Cluster Computing, 2007 IEEE International Conference on*, 129 –138.