5-15-2012

EVALUATING SCHEDULING METHODS FOR ENERGY COST REDUCTION IN A HETEROGENEOUS DATA CENTER ENVIRONMENT

Tobias Brandt
University of Freiburg

Christian Bodenstein
University of Freiburg

EVALUATING SCHEDULING METHODS FOR ENERGY COST REDUCTION IN A HETEROGENEOUS DATA CENTER ENVIRONMENT

Brandt, Tobias, University of Freiburg, Chair of Information Systems Research, Platz der Alten Synagoge, 79085 Freiburg, Germany, tobias.brandt@is.uni-freiburg.de

Bodenstein, Christian, University of Freiburg, Chair of Information Systems Research, Platz der Alten Synagoge, 79085 Freiburg, Germany, christian.bodenstein@is.uni-freiburg.de

Abstract

Over the past two decades the rise of information technologies (IT) has enabled businesses to communicate, coordinate, and cooperate in unprecedented ways. However, this did not come without a price. Today, IT infrastructure accounts for a substantial fraction of the national energy consumption in most advanced countries. Subsequently, research turned to finding ways of making IT more sustainable and lessening the environmental impact of IT infrastructure.

In our previous work we developed LINFIX, an innovative method for handling the scheduling problem in data centers, which substantially reduced the total energy consumption compared to commonly used practices. Due to the computational complexity of the scheduling problem, we were, however, unable to estimate the cost reduction of LINFIX compared to what is theoretically possible.

In this work we employ a genetic algorithm to provide a benchmark to better assess the quality of the LINFIX solutions. While the genetic algorithm frequently finds better solutions, the additional average cost reduction when compared to LINFIX is less than 0.1 percent. Taking the computational speed into account, this confirms our hypothesis that LINFIX provides very energy efficient scheduling plans in short time.

Keywords: Green IT, Sustainability, Genetic Algorithm, Data Center Scheduling.

1 Introduction

During the past two decades data centers have turned into an integral part of the modern business environment. The functions they provide have become essential to economic, scientific, and technological organizations. However, this growth did not come without a price. The U.S. Environmental Protection Agency (2007) estimated that the energy use of U.S. servers and data centers doubled between 2000 and 2006, reaching about 61 billion kW/h or 1.5 percent of the national electricity consumption in the final year. Additionally, the rise of regenerative energies, the effects of the global climate change, and a growing desire to become independent from nuclear energy have shifted public interest towards using energy in a more sustainable manner. Taking into account the substantial energy consumption caused by IT infrastructure, one way to support achieving this goal is by reducing the energy demand of data centers.

The Industrial Technologies Program located at the U.S. Department of Energy (2011) had partnered with the industry to decrease data center power consumption by 10% by 2011. The results from these efforts are yet to be determined. Their focus lies heavily on updating server hardware, cooling and infrastructure systems to more energy efficient levels. These measures are usually associated with high implementation costs and, while these investments may pay back within one or two years, they are still difficult to bear for small- and medium-sized companies.

A different approach to reducing electricity demand is analyzing the way how workloads are allocated to the specific servers in terms of energy efficiency, as data centers are frequently comprised of servers that are somewhat heterogeneous with respect to capacity and power consumption. This heterogeneity can derive from at least three causes: (1) Failed components are replaced with newer and better ones, (2) necessary increases in performance or capacity favor more powerful components, as well, and (3) the emergence of blade servers which are more space- and energy efficient than their PC-style predecessors (Heath et al. 2005). Finding the most energy efficient allocation not only lessens the environmental impact of high power consumption, but is also in the best economic interest of data centers. As energy costs are taking up an increasing share of the company budget and the costs to power servers may already exceed the purchasing costs (Poess and Nambiar 2008), effective ways to reduce power consumption decide about survival or death in an increasingly competitive market environment.

Bodenstein et al. (2011) have shown that an exact solution to these nonlinear scheduling problems can take up to several minutes even for relatively small problems and very frequently the optimal solution takes up hours to calculate. Since a decision is usually needed within a few seconds or less, approximation methods and heuristics that yield good, if not necessarily optimal, solutions are used instead. Vengerov (2009) mentioned the Best-Fit algorithm as a reasonably good method commonly used by data centers to allocate workloads to resources. This claim was verified by Bodenstein et al. (2011), who also compared the performance of Best-Fit and several other methods. While Best-Fit allows for very fast allocation decisions, it was outperformed by several approximation methods in a heterogeneous server environment, which yielded less cost-intensive solutions. One method, LINFIX, produced significantly better results at a negligible increase in computational cost. However, due to the computational complexity of the nonlinear problem, it proved to be difficult to compare these results to the exact results of the nonlinear optimization problems for a large data set, and thereby gauge the loss from approximation.

In this work we employ a genetic algorithm to better assess the overall performance of the LINFIX method in producing energy efficient schedules to allow for a more sustainable management of data centers. Although the genetic algorithm is a metaheuristic and as such not guaranteed to produce the optimal solution to the nonlinear scheduling problem, it provides valuable contributions:

- It produces a new benchmark composed of consistently equal or better results for an improved performance assessment of the LINFIX method

- It describes a new avenue for schedule optimization in data centers
- It illustrates potential savings in energy expenditures for data center manager

The following section will provide an overview on work related to the problem at hand. This is followed by a section on the models used in this paper. Section four illustrates the genetic algorithm developed to fit these models, while section five describes the data generation. Section six presents and evaluates the results of the simulations. Section seven concludes.

2 Related Work

Recently, there have been several research approaches to optimizing costs and profits in a data center environment. The most obvious route is through updating power inefficient systems, but as has been previously mentioned, it is not always the most affordable one. Patel and Shah (2005) propose a comprehensive cost model for the planning, development and operation of a data center and explore the costs and benefits of "smart" heat control, building on work by Beitelmal and Patel (2002). This has led to further research in data center architecture and thermal management (e.g. Bash et al. 2006, Lim et al. 2008), as well as research into the gains that can be realized by unifying workload, power and cooling management (Wang et al. 2010).

Approaches towards the optimization of data center operations by intelligently allocating workloads to servers have been twofold: Firstly, maximizing profit in case of oversaturated resources, implying a selection of jobs to be accepted and to be denied. This causes the possibility of having to pay penalties if certain jobs cannot be accepted or have to be canceled. Secondly, minimizing energy costs for undersaturated data centers, where basically all jobs can be accepted and the only question is what job to run on which server. The first case exhibits active constraints on resources on the supply side, leading naturally to more market-based and auction theoretical solutions to the optimization problem (Buyya and Murshed 2001; Chun and Culler 2002). However, Burge et al. (2007) note, that most data centers are undersaturated, as their capacity is designed to handle peak loads. Nathuji et al. (2008) describe how the development and improvement of virtualization software reduced the need to allocate resources to applications based on peak load characteristics to guarantee performance, thus making these resources available to other tasks. These results strengthen the case for the second approach, which seeks to allocate workloads such that the operating costs of servers are minimized, ignoring the question of resource feasibility.

The question remains how much energy can be saved by these allocation mechanisms. Moore et al. (2005) describe the example of the HP Proliant DL360 G3 server that has a measured power consumption of 150W when idle and 285W when running at full utilization. This is not an exceptional case as most servers consume up to 60% of their power consumption at maximum utilization even when idle (Fan et al. 2007). For that reason Burge et al. (2007) have included the possibility to turn idle servers off into their models. However, since completely turning off machines poses the possibility of failure when booting them back up and frequently idle periods have a time span of less than one second, Meisner et al. (2009) propose with PowerNap and RAILS two efficient measures to significantly reduce idle server power consumption without the aforementioned problems. The availability of measures like PowerNap is crucial to reap the benefits from intelligent job allocation mechanisms. A consequence of this is that it is frequently optimal to fully utilize a server before booting another one. This is also the aim of the Best-Fit method, which allocates jobs such that servers are as close to their maximum utilization as possible.

Building on these results there has been some research on "spatio-temporal thermal-aware" optimization (Mukherjee et al. 2009; Tang et al. 2008), which allocates jobs according to a model that not only includes power required by servers to process their workloads, but also heat and cooling effects in the physical server environment. These are also among the few cases where genetic algorithms have been used in this context. The problem with this approach is that it requires extensive knowledge about the physical environment and thermal interactions in the specific server room, which

again raises the issue of affordability. Aside from these cases there has been very little use of genetic algorithms in the context of optimal scheduling in data centers, although they have been successfully adapted to a wide array of problems ranging from control engineering (Li et al. 1996) to power electronic circuit optimization (Zhang et al. 2006) and the selection of fuzzy rules in classiffication problems (Ishibuchi et al. 1995).

3 Model Structure

Bodenstein et al. (2011) proposed several approximation methods for an underlying nonlinear model that relates node utilization to energy costs. The objective function of this model is illustrated below:

$$\min_x V = \sum_{n \in N} \sum_{t \in T} \left[a_n \left(\frac{\sum_{j \in J} x_{jnt} cd_j}{cs_n} \right)^{b_n} + c_n y_{nt} \right] \quad (1)$$

LINFIX, an evidently well performing approximation method, leaves the fixed cost component unchanged, but reduces the variable cost component to a linear function. This results in an objective function as follows:

$$\min_x K = \sum_{n \in N} \sum_{t \in T} \left[varco_n \left(\frac{\sum_{j \in J} x_{jnt} cd_j}{cs_n} \right) + kfix_n y_{nt} \right] \quad (2)$$

V and K represent the Total cost according to the nonlinear, respectively the LINFIX, objective function with a_n, b_n, c_n being the exogenous cost function parameters of node n and $varco_n$ and $kfix_n$ being the parameters for linear approximation of cost function of node n. Θ_0 is the Heaviside Step Function with $\Theta_0[0] = 0$. cs_n and ms_n are the computing and memory units supplied by node n.
The necessary information about each order is provided by cd_j, the computing units required by order j, md_j, the memory units required by order j, as well as $first_j$ and $last_j$, which describe the first and final periods of order j.
The actual allocations are determined by the states of all x_{jnt}, which allocates each job j to node n in period t. Subsequently, y_{nt} determines for each node n at period t if it is switched on or off.
Both functions are subject to the following constraints:

$$x_{jnt} \in \{0,1\} \qquad \forall j \in J, \forall n \, E \, N, \forall t \in T \quad (3a)$$

$$\sum_{n \in N} x_{jnt} = \begin{cases} 1 & \forall j \in J, \forall t \in [first_j, last_j] \\ 0 & \forall j \in J, \forall t \in T \setminus [first_j, last_j] \end{cases} \quad (3b)$$

$$\sum_{j \in J} x_{jnt} cd_j \leq cs_n \qquad \forall n \, E \, N, \forall t \in T \quad (3c)$$

$$\sum_{j \in J} x_{jnt} md_j \leq ms_n \qquad \forall n \, E \, N, \forall t \in T \quad (3d)$$

$$y_{nt} \in \{0,1\}, \quad \Theta_0\left[\sum_{j \in J} x_{jnt}\right] = y_n t \qquad \forall n \, E \, N, \forall t \in T \quad (3e)$$

N describes the set of nodes in the scheduling problem, J the set of jobs and T the set of time periods being considered. The objective function, equation (1), attempts to minimize electricity costs over all time periods T and all nodes N. Two central assumptions are that there is no intertemporal linkage, such that it is irrelevant in period t if the server was switched on or off in period $t - 1$; and that jobs are fully migratable, i.e. it does not incur additional migration costs if job j switches from node n_1 in period $t - 1$ to node n_2 in period t. In accordance with the findings on PowerNap and advances in virtualization techniques, both assumptions are quite reasonable in most real world settings.
In the nonlinear model the electricity costs are captured by an allometric cost function of the form $Y = aX^b + c$. X is in this case the ratio between the current utilization $\sum_{j \in J} x_{jnt} cd_j$ and the maximum

utilization cs_n of the specific node. The fixed component c only activates when node n is used at all in that time period, since y_{nt} will be zero, otherwise.
The objective function for the LINFIX model, K, simplifies the allometric function to a linear approximation of that function. Its parameters $varco_n$ and $kfix_n$ are chosen such that the resulting line fits best to describe the original allometric function with the parameters a_n, b_n and c_n.
Both models make the implicit assumption that the resources provided by any node n are perfectly divisible, which can again be attained through virtualization software up to a certain degree. A further assumption that restricts possible solutions is that jobs are indivisible, i.e. a job cannot be split onto two or more nodes.
The constraints come quite naturally. Constraint (3a) enforces the boolean nature of x_{jnt}. A job is either allocated to a specific node at a specific time or it is not. Equation (3b) assures that each job is being allocated to exactly one node in those time slots where it is to be executed and to exactly zero nodes in those time slots where it is not. Thus it checks for the requirements that all jobs have to be accepted and that all jobs are indivisible. Constraints (3c) and (3d) check that each node is not tapped for more computing and memory resources than it can provide. Finally, constraint (3e) can be interpreted as an on/off-switch for each node. As soon as any job is allocated to node n it is equal to one (and, consequently, the fixed cost component for that node applies), otherwise it is equal to zero.

Bodenstein et al. (2011) have shown that the LINFIX method can find optimal solutions to its objective function about as quickly as other common approximation methods with an improvement to the quality of the solutions. However, the cost increase incurred by any form of approximation has been found to be hard to measure as searching for the exact solution to the true function V takes common nonlinear solvers a tremendous amount of time, if they find a solution at all. For that reason we resort to a genetic algorithm to locate near-optimal solutions, which is described in the following section.

4 A phased genetic algorithm

The concept of genetic algorithms was originally introduced by Holland (1975). It transfers the notions of selection, mating, and mutation from evolution to computer science. A comprehensive introduction can be found in Goldberg (1989).
The scheduling problem described in the previous section can be reduced to the question of which job is allocated to which node at which time. This question is redundant for jobs that are not active at a certain time period, since they do not need to be allocated. Henceforth, only the jobs that are actually active at a given time need to be considered and encoded. Eventually this lead to a coding as illustrated in figure 1. In this example case there are five different jobs, of which jobs 1 and 5 are active during the first period, jobs 1, 2, 4, and 5 during the second period, and so forth as depicted. They can be allocated to a choice of three different nodes – 0, 1, or 2. It can be seen that the coding of the solution immediately reflects the allocation decision with both jobs in period 1 being allocated to node 0, all jobs in period 2 being allocated to that node as well, etc. In the simulations cases with up to 50 nodes were considered. To keep the easy interpretation of solution strings even for those cases the algorithm uses a base $N + 1$ numerical system (N being the number of nodes in the scheduling problem), thus allowing a single digit to represent each node in the problem. This includes the digits 0 to 9, after which the lower case Latin standard alphabet and, subsequently, the upper case Latin standard alphabet is included.
Early trials with a simple genetic algorithm just including the operators selection, crossover, and mutation yielded unsatisfying results even when very large population sizes and long run times were allowed. The reason for this is believed to be the high proportion of fixed costs in relation to total costs in the model. Consequently, it is frequently optimal to fully utilize a server before booting another one. Even the simple form of the genetic algorithm performed well in that regard and quickly stacked jobs onto only a small number of nodes. However, it was very prone to getting stuck at local optima, since crossover and mutation did not provide the sufficient ability to escape from these. For that reason a number of additional operators were introduced. These included inversion, in which either an

entire segment of the string is flipped (string-segment inversion) or just the endpoints of that segment are switched (endpoint inversion). In addition to that two more systematic approaches to mutation were introduced.

Job	15	1245	12346	134
Node	00	0000	02002	000
t	1	2	3	4

Figure 1. Coding of solutions in the genetic algorithm

The first type of systematic mutation operator assumes that the structure of the current solution, i.e. the subsets of active jobs that are allocated to the same node, is optimal. For example, the solutions *iijiiii*, *ccjcccc* and *ii7iiii* would all have the same structure, since the active jobs {1; 2; 4; 5; 6; 7} and the active job {3} are always allocated to one node, respectively. Contrary to that, the string *jiiiiii* would not have the same structure. The operator then randomly selects a position within the string and changes the value at that position, as well as at all positions that shared this node, to a different new value. So, if the second position in the string 111222333 would be selected and the new node after mutation would be 4, then position 2 would change to 4, as well as all positions that originally shared a node with position 2 – resulting in the string 444222333. This operator allows the genetic algorithm to move between different locally optimal values with just one operation. It is a mechanism to escape from local optima, a feat that was unlikely to be achieved by the standard operators.

However, it is not always the case that all good solutions share a common structure. Hence, the second type of systematic mutation checks the current optimal solution for the maximum number of jobs on any node, *m*. This gives the algorithm an idea what a good stack size for this problem might be, i.e. the number of jobs commonly allocated to a single node. Next, a node is randomly chosen and each position in the string is mutated to this node with a probability of $\frac{m}{J_t^a}$, i.e. the pile size over the string length as J_t^a is the number of active jobs at time *t*. This directly implies that the expected number of jobs that are allocated to this node after mutation is equal to m, given that no jobs are allocated to this node before mutation. However, there is a high variance and anything from zero to all positions could mutate. Essentially this operator is very similar to the classical mutation operator with the exception that it exploits the fact that all optimal solutions tend to fully utilize nodes before activating new ones. Opposed to the type I systematic mutation it also allows the genetic algorithm to explore solutions with an underlying structure different from the currently best solutions.

The problem created by the inclusion of these operators was that the population is changed so thoroughly every generation that the point of the genetic algorithm gets lost. This can be alleviated either by decreasing the probabilities for each operator to be applied or by separating the algorithm into different phases. For the algorithm in this paper the second method was used as it provided very good results. The algorithm is separated into three phases which only differ in terms of which operators are applied to the strings in the population. The first phase of the algorithm is equivalent to a simple genetic algorithm consisting of selection, crossover, and mutation, with the exception that endpoint inversion has been included. The purpose of this phase is to find some locally optimal allocations in the early stages and to tune allocations in later stages more finely. The second phase applies the high-impact operators. Generations alternate between applying crossover and applying one of type I or type II systematic mutation, string segment inversion, or no operator, at all. This ensures that a selection process occurs after each operator and crossover takes place between solutions produced by a wide array of powerful operators. The final phase is a duplicate of the simple genetic algorithm with a higher mutation rate. This ensures that the population keeps a sufficient degree of diversity, which otherwise could cause issues due to the high frequency of the selection operator in the second phase. A full cycle of all three phases lasts 25 generations (10, 11, and 4 for each phase, respectively), after which the algorithm returns to the first phase. This is illustrated in figure 2.

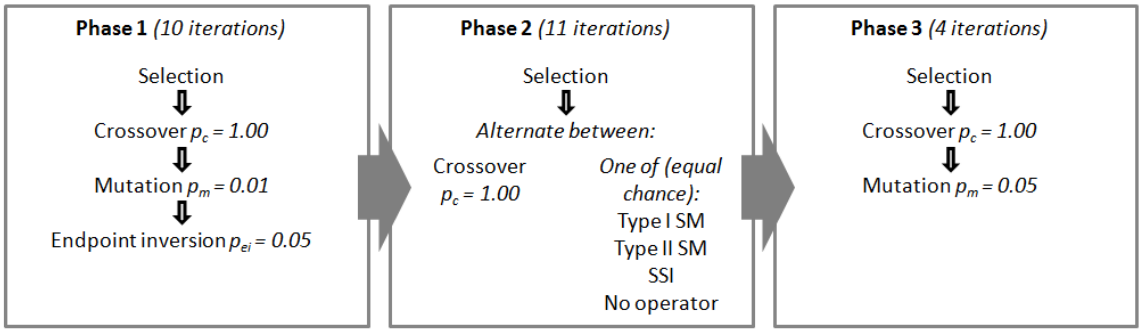

Figure 2. Schematic summary of the three-phase genetic algorithm (SM = systematic mutation, SSI = string segment inversion)

Finally, it should be mentioned that the selection method used is fitness proportionate selection, with the solution incurring the lowest cost getting the highest probability of being selected. The crossover operator uses simple one-point crossover.

5 Data Generation

The data set used in this paper has been created by a scenario generator similar to the one used in Bodenstein et al. (2011). The nodes are randomly selected from a list of 182 common server systems according to results from the Standard Performance Evaluation Corporation, which provide performance-to-power-ratios and other data related to power consumption for each type of server (SPEC, 2008). The generator then derives the parameters for the allometric cost function and the number of CPUs of this specific type of system. It is assumed that on average 10 servers are aggregated to a single node, such that the total of available resources for a specific node is equal to 10 times the number of CPUs in the system it is comprised of. This led to node sizes ranging between 20 and 480 CPUs with an average of 91 and a median of 80 CPUs. Since the purpose of this paper is a comparison between LINFIX results and the best solutions given by the genetic algorithm without an extensive runtime analysis, there was no variability with respect to the time horizon and the resource factor. The allocation time frame was set to 4, i.e. jobs could last either 1, 2, 3, or 4 time periods. The resource factor was set to 0.25, implying that the expected job size is equal to one quarter of the median node size. This resulted in uniformly distributed job sizes between 11 and 30 CPUs.
The number of jobs per scenario ranged between 5 and 40, while the number of nodes ranged between 5 and 50, both in steps of 5. This resulted in 80 possible scenarios. For each of these combinations 10 instances were generated, totaling to 800 instances overall.
Finally, the time frame for each specific job was randomly chosen according to a discrete uniform distribution between the four periods for the starting period, and between the starting period and all later periods for the final period.

For each instance the solutions according to the LINFIX method were produced. This was done using the GAMS/CPLEX solver and yielded 790 results, which would then serve as a benchmark for the solutions yielded by the genetic algorithm. These allocations optimized the simplified linear model; the remaining 10 scenarios were shown to be infeasible.

6 Simulation Results

Table 2 shows the average difference between the results produced by LINFIX and those yielded by the genetic algorithm in percent for each combination of jobs and nodes. For example, over all instances with 20 jobs and 20 nodes the LINFIX result was on average 0.17 percent higher than the one produced by the genetic algorithm. Table 3 reduces the sample to those cases where the algorithm

provided a better result than LINFIX. Thus, it shows the average difference between the respective results conditional on the genetic algorithm finding a better solution. So, over all instances with 20 jobs and 20 nodes, if the genetic algorithm found a better solution than LINFIX, the solution given by LINFIX incurred on average a cost 0.3 percent higher than that solution.

Over the set of all instances the genetic algorithm found a better solution than LINFIX in 342 cases or 43.29 percent. If the genetic algorithm found a better solution, the LINFIX solution had on average a 0.22 percent higher cost associated with it. If the cases where both methods found the same solution are included, the solutions given by LINFIX incurred on average a 0.09 percent higher total cost.

Considering the numbers shown in table 2 and table 3 the runs performed by the genetic algorithm strongly confirm a high quality of the solutions provided by the LINFIX approximation. Results diverged only on a few occasions by more than one percent, and over all instances the difference in total cost incurred amounted to less than a tenth of a percent.

J \ N	5	10	15	20	25	30	35	40	45	50
5	0	0.05	0.03	0.04	0.08	0.03	0.09	0.12	0.05	0.18
10	0.03	0.01	0.03	0.18	0.07	0	0	0.01	0.21	0.13
15	0.01	0.06	0.29	0.05	0.12	0.11	0.18	0.05	0.21	0.06
20	0.03	0.05	0.1	0.17	0.09	0.07	0.21	0.19	0.06	0.28
25	0.2	0.2	0.08	0.06	0.09	0.06	0.1	0.02	0.09	0.06
30	0.09	0.07	0.13	0.16	0.16	0.03	0.1	0.07	0.07	0.21
35	0.07	0.08	0.12	0.07	0.03	0.13	0.16	0.01	0.08	0.16
40	0.16	0.06	0.08	0.08	0.05	0.03	0.04	0.18	0.08	0.07

Table 2. Average difference between LINFIX and the genetic algorithm results (in percent)

J \ N	5	10	15	20	25	30	35	40	45	50
5	N/A	0.13	0.13	0.31	0.47	0.33	0.42	0.63	0.24	0.46
10	0.24	0.05	0.09	0.45	0.21	N/A	N/A	0.13	0.73	0.43
15	0.06	0.29	0.46	0.15	0.42	0.42	0.25	0.19	0.34	0.16
20	0.12	0.1	0.24	0.3	0.22	0.15	0.5	0.36	0.19	0.34
25	0.63	0.36	0.14	0.12	0.12	0.14	0.2	0.04	0.12	0.12
30	0.43	0.12	0.17	0.18	0.33	0.09	0.16	0.12	0.11	0.3
35	0.29	0.1	0.24	0.16	0.04	0.22	0.19	0.17	0.13	0.22
40	0.58	0.13	0.09	0.09	0.14	0.06	0.05	0.23	0.11	0.15

Table 3. Average difference conditional on the genetic algorithm finding a better solution (in percent)

	t=2	t=3
Genetic Algorithm	8888877	7777888888
LINFIX	7777788	5555555555

Figure 3. Different schedules yielded by the LINFIX method and the genetic algorithm

The highest relative difference between the results produced by LINFIX and the genetic algorithms was found in a scenario with 15 job and 15 nodes – LINFIX yielded a total cost of 968.14 MU (monetary units), while the solution provided by the genetic algorithm only incurred a cost of 950.10 MU. This amounts to a difference of 1.90 percent. Both methods produced the same allocation for the period 1 and period 4; however, they diverged for the remaining two periods. Figure 3 illustrates these differences. The nodes included in these allocations are 7, 8, and 5. The nonlinear and linear cost equations associated with these nodes are presented below.

$$C_{nlin,5}(x_5) = 339.16 \cdot \left(\frac{x_5}{480}\right)^{0.81} + 157.11 \quad (4a)$$

$$C_{lin,5}(x_5) = 329.09 \cdot \left(\frac{x_5}{480}\right) + 178.27 \quad (4b)$$

$$C_{nlin,7}(x_7) = 103.49 \cdot \left(\frac{x_7}{120}\right)^{1.01} + 74.9 \quad (5a)$$

$$C_{lin,7}(x_7) = 103.61 \cdot \left(\frac{x_7}{120}\right) + 74.58 \quad (5b)$$

$$C_{nlin,8}(x_8) = 112.01 \cdot \left(\frac{x_8}{120}\right)^{0.72} + 58.84 \quad (6a)$$

$$C_{lin,8}(x_8) = 106.48 \cdot \left(\frac{x_8}{120}\right) + 69.75 \quad (6b)$$

$C_{nlin,n}$ and $C_{lin,n}$ are the nonlinear and linear cost functions for a single period for node n, respectively, while x_n is the total workload allocated to that node in that period.

There are some striking differences between the nonlinear cost functions. $C_{nlin,7}(x_7)$ is very close to linearity, while the other two functions are quite the opposite, illustrated by exponents substantially smaller than 1. In period $t = 2$ the total cost according to the nonlinear functions are 274.22 MU for the genetic algorithm and 281.62 MU for LINFIX. Contrast that to 279.26 MU and 277.27 MU, respectively, if the linear approximations are used. It should not come as a surprise that these differences are mostly caused by node 8 and its strongly nonlinear cost function. The genetic algorithm allocates a total workload of 118 to node 8 and of 35 to node 7, while LINFIX flips these allocations. At a workload of 35 the linear approximation underestimates the true cost, while at a workload of 118 it overestimates it. Since the other node in the allocation, 7, is almost linear resulting in a very good linear approximation, it cannot compensate this error and LINFIX chooses the wrong schedule. The same reasoning applies to period $t = 3$. Here the genetic algorithm allocates a workload of 74 to node 7 and of 120 to node 8, fully utilizing it. LINFIX stacks all jobs on the high-capacity node 5. Thereby it again overestimates the costs incurred on node 8 and underestimates those on node 5, resulting in a worse allocation.

Summarizing this analysis the following behavior of the LINFIX method can be noted. Assuming (1) a high nonlinearity of the cost functions of the active nodes and (2) a total workload that is just slightly below the sum of the workloads the currently activated nodes can handle, then LINFIX will be biased to allocate a higher than optimal workload onto those nodes whose nonlinear cost functions have a larger exponent. Consequently, LINFIX works very well for cost functions that are almost linear or those cases where all active nodes exhibit a similar degree of nonlinearity, such that no particular node is preferred due to the bias.

Figure 4 illustrates the runtime analysis for both methods. Evidently, LINFIX solves all tested combinations of nodes and jobs in less than 10 seconds, while the runtime of the genetic algorithm quickly exceeds several minutes. However, the genetic algorithm was designed in a way to provide as close to optimal solutions as possible. Therefore the exit condition was set for the genetic algorithm to run for several hundred generations without improvement of the best solution before it terminates.

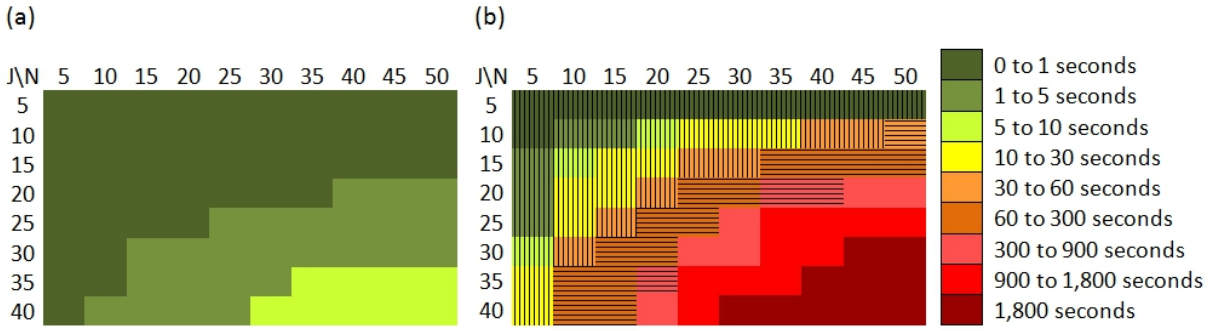

Figure 4. Runtime analysis for LINFIX (a) and the genetic algorithm (b)

Nevertheless, the genetic algorithm was also programmed such that it documents the currently best solution in each generation. Thus, we were able to derive when the algorithm actually arrived at its final result. These results are indicated by the vertical and horizontal lines in figure 4 (b). Vertical lines represent that the solution has been found in less than 10 seconds, while horizontal lines represent that it has been found in less than 60 seconds. Depending on the time constraint at hand this indicates that the genetic algorithm is actually a valid scheduling mechanism by itself, at least for certain combinations of jobs and nodes. Nevertheless, the results in figure 4 confirm that LINFIX is capable of quickly deriving excellent results.

7 Conclusion

In this paper we contrasted two innovative methods for addressing the scheduling problem in data centers while reducing overall energy consumption. One, LINFIX, provided the exact solution to a linear approximation of the nonlinear scheduling problem, while the other, the genetic algorithm, produced an approximated solution to the exact nonlinear problem.
We simulated both methods over a set of 790 different instances and derived two central results:

- Compared to other solving methods, LINFIX provides excellent schedules at a very low computational cost. Over all instances it incurs an increased energy consumption of less than 0.1 percent compared to the solutions of the genetic algorithm.
- Depending on the time constraint, the genetic algorithm is a valid scheduling method for certain scheduling problems.

Both methods provide schedules that substantially decrease energy consumption compared to commonly used practices. This enables the utilization of energy in a more sustainable manner by decreasing the overall energy demand of the IT infrastructure.

Data center managers face an additional incentive to implement one of the presented methods, since the reduction of energy demand is directly associated with a reduction of energy-related costs. Beyond these direct practical implications, this work contributes to the research on efficient scheduling by juxtaposing two distinct scheduling methods and comparing the quality of the solutions provided and the associated runtime. We show that there is no clearly superior method, since either method, LINFIX or the genetic algorithm, may be preferable depending on the circumstances.

In our future research we will investigate potential improvements to the genetic algorithm. The first prototype used in our simulations included operators that were specifically adapted to the problem at hand. However, there may still be ways to improve these operators and increase the computational speed of the algorithm. We also plan to include intertemporal linkage in our problem design to reflect possible gains from using the same node over several time periods.

8 References

Bash, C.E., Patel, C.D., and Sharma, R.K. (2006). Dynamic thermal management of air cooled data centers. In Proceedings of ITHERM.

Beitelmal, A. and Patel, C. (2002). Thermo-Fluids Provisioning of a High Performance High Density Data Center. Hewlett Packard Technical Report HPL 2004-146 (R.1), Hewlett-Packard Laboratories Palo Alto.

Bodenstein, C., Schryen, G., and Neumann, D. (2011) Reducing datacenter energy usage through efficient job allocation. In Proceedings of the 19th European Conference on Information Systems - ECIS 2011 paper 108

Burge, J., Ranganathan, P., and Wiener, J. (2007). Cost-aware scheduling for heterogeneous enterprise machines (CASH 'EM). In Cluster Computing, 2007 IEEE International Conference on, pp. 481 – 487.

Buyya, R. and Murshed, M.M. (2001). A Deadline and Budget Constrained Cost-Time Optimisation Algorithm for Scheduling Task Farming Applications on Global Grids. In Int. Conf. on Parallel and Distributed Processing Techniques and Applications, Las Vegas.

Chun, B. and D. Culler (2002). User-Centric Performance Analysis of Market-Based Cluster Batch Schedulers. In Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on, p. 30.

Fan, X., Weber, W.D., and Barroso, L.A. (2007) Power provisioning for a warehouse-sized computer. In ACM SIGARCH Computer Architecture News. doi: 10.1145/1250662.1250665

Goldberg, D. (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley Longman, Bonn

Heath, T., Diniz, B., Carrera, E.V., Meira Jr., W., and Bianchini, R. (2005) Energy conservation in heterogeneous server clusters. In Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming. doi: 10.1145/1065944.1065969

Holland, J.H. (1975) Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor

Ishibuchi, H., Nozaki, K., Yamamoto, N., and Tanaka, H. (1995) Selecting fuzzy if-then rules for classification problems using genetic algorithms. In IEEE Transactions on Fuzzy Systems. doi: 10.1109/91.413232

Li, Y., Ng, K.C., Murray-Smith, D.J., Gray, G.J., and Sharman, K.C. (1996) Genetic algorithm automated approach to the design of sliding mode control systems. In International Journal of Control. doi: 10.1080/00207179608921865

Lim, K., Ranganathan, P., Chang, J., Patel, C., Mudge, T., and Reinhardt, S. (2008). Understanding and Designing New Server Architectures for Emerging Warehouse-Computing Environments. In Computer Architecture, 2008. ISCA '08. 35th International Symposium on, pp. 315 – 326.

Meisner, D., Gold, B.T., and Wenisch, T.F. (2009) PowerNap: eliminating server idle power. In Proceedings of the 14th international conference on Architectural support for programming languages and operating systems. doi: 10.1145/1508244.1508269

Nathuji, R., Isci, C., Gorbatov, E., and Schwan, K. (2008). Providing platform heterogeneity-awareness for data center power management. In Cluster Computing 11.

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 5—5

Mukherjee, T., Banerjee, A., Varsamopoulos, G., Gupta, S.K., and Rungta, S. (2009). Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers. In Computer Networks 53 (17).

Patel, C. D. and Shah, A.J. (2005). Cost Model for Planning, Development and Operation of a Data Center. Hewlett Packard Technical Report HPL-2005-107(R.1), Internet Systems and Storage Laboratory, HP Laboratories Palo Alto.

Poess, M. and Nambiar, R.O. (2008) Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results. In Proceedings of the VLDB Endowment. doi: 10.1145/1454159.1454162

SPEC (2008) SPECpower_ssj2008 benchmark. http://www.spec.org/specpower/

Tang, Q., Gupta, S.K.S., and Varsamopoulos, G. (2008). Energy-Efficient Thermal-Aware Task Scheduling for Homogeneous High-Performance Computing Data Centers: A Cyber-Physical Approach. In IEEE Transactions on Parallel and Distributed Systems 19.

U.S. Department of Energy, Industrial Technologies Program (2011) Saving Energy in Data Centers. http://www1.eere.energy.gov/industry/datacenters/index.html. Accessed 08 March 2011

U.S. Environmental Protection Agency (2007) Report to Congress on Server and Data Center Energy Efficiency – Response to Public Law 109—431. http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf. Accessed 08 March 2011.

Vengerov, D. (2009) A reinforcement learning framework for utility-based scheduling in resource-constrained systems. In Future Generation Computer Systems. doi: 10.1016/j.future.2008.02.006

Wang, Z., Tolia, N., and Bash, C. (2010). Opportunities and challenges to unify workload, power, and cooling management in data centers. In SIGOPS Oper. Syst. Rev. 44.

Zhang, J., Chung, H., and Lo, W. (2006) Pseudocoevolutionary genetic algorithms for power electronic circuits optimization. In IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews. doi: 10.1109/TSMCC.2005.855497