Constrained Optimizaton Modelling on Resource Allocation for Distributed Systems

Teng Yu

Fall 2015

Abstract

The abstract is a very brief summary of the report's contents. It should be about half page long. Somebody unfamiliar with your project should have a good idea of what it is about having read the abstract alone and will know whether it will be of interest to them.

This work intends to improve the performance of resource allocation process on heterogeneous distributed systems by designing a novel allocation model. It provides a novel approach to involve order theory to model the allocation process and constraints by research on the relations between different resource allocation requests(RAr) and mapping with the structure inside the cluster of servers, then designing ranking functions on their topology as a metric for optimisation instead of viewing the underlining problem as a multi-dimensional bin-packing instance and framing the process as a special Linear Programming formulation which is common in the literature. It focuses on the modelling process and invokes well-known algorithms in experiment to compare its performance with other models. Finally, we discuss that this approach can be easily extended to apply on not only the fields within distributed systems (such as for cloud computing platform), but also in the wide fields of computer science such as for Information Retrieval and HPC.

Acknowledgements

It is usual to thank those individuals who have provided particularly useful assistance, technical or otherwise, during your project. Your supervisor will obviously be pleased to be acknowledged as they will have invested quite a lot of time overseeing your progress.

Sincere thanks to Dr.Mark Stillwell for his countless helpful comments and advise during this project. Thanks Prof.Alexander Wolf for his supervision. Thanks my personal tutor Dr.Sadri Fariba and my friends in Imperial College for their helps.

Contents

This should list the main chapters and (sub) sections of your report. Choose self-explanatory chapter and section titles. If possible you should include page numbers indicating where each chapter/section begins. Try to avoid too many levels of subheading. Try if possible to stick to sections and subsections; subsubsections are usually avoidable.

1 Introduction

This is one of the most important components of the report. It should begin with a clear statement of what the project is about so that the nature and scope of the project can be understood by the reader. It should summarise everything you set out to achieve, provide a clear summary of the project's background and relevance to other work, and give pointers to the remaining sections of the report that contain the bulk of the technical material.

2 Background and Related Work

The background and related work section of the report should set the project into context by relating it to existing published work that you read at the start of the project when your approach and methods were being considered. There are usually many ways of approaching a given problem, and you should not *just pick one at random. Describe and evaluate as many alternative approaches as possible. The published work may be in the form of research papers, articles, text books, technical manuals, or even existing software or hardware of which you have had hands-on experience. Do not be afraid to acknowledge the sources of your inspiration; you are expected to have seen and thought about other people's ideas, so your contribution largely will be putting them into practice in some other context.*

3 Body of report-Original Ideas

The central part of the report typically consists of three of four chapters detailing the technical work undertaken during the project. The structure of these chapters is highly project dependent. Usually they reflect the chronological development of the project, e.g., design, implementation, experimentation, and optimisation, although this is not always the best approach. However you choose to structure this part of the report, you should make it clear how you arrived at your chosen approach in preference to the other alternatives documented in the background. For implementation projects you should describe and justify the design of your system at some high level, for example by using any of the design methods taught during the first- and second-term courses, and should document any interesting problems with, or features of, your implementation. Integration and testing are also important to describe. Your supervisor will advise you on the most suitable structure for these middle sections.

4 Basic RArs

In this section, we discuss the relation between basic resource allocation requests (RArs). We assume RArs are independent between different dimensions in the theoretical case and focus on the relation of RArs in each dimension to build up their foundation model.

4.1 RArs Relations

Consider to handle with resource that have difficult-to-represent capacities. User's request may be special on different processing element (heterogeneous)or say the relation between them. Then we find the capacity of a cluster is the set of all allocation request that it can service. Try to give a relation between RArs:

Definition 1 *Given the relation $\preceq$ between RAr:*

$$\forall \alpha \in a\ cluster\ device, \exists A, B \in RAr, such\ that\ A \preceq B\ \Leftrightarrow \alpha(B) \mapsto \alpha(A).$$

We say $\alpha(B)$ is true iff α can service B. Then we can define a parallel relation between RArs as follow:

Definition 2

$$\forall A, B \in RAr, A \npreceq B \ \wedge B \npreceq A \ \Leftrightarrow A \parallel B.$$

Followed by defining relations between user's requests, we can try to build a closed topology involving those requests. It will be more likely a lattice as we can ordered those requests by inclusion through the relation we just defined. As the topology inside a cluster device can be viewed as a littice as well by considering that the intersection of subset servers in cluster, it leads to a possibility to achieve mappings during the allocating process.

A trivial approach from the order theory is to consider Birkhoff's representation theory, which says modular lattice can be represented by down-sets of its corresponding partial order composed by join-irreducible elements [2]. Then it is easy to view the RArs as downset of the lattice of cluster topology. Then it remains to create a metric or say ranking function to compare the performance between different mapping.

4.2 RArs Model

Consider there is a common foundation of all RArs which is a non resource needed request, we represent it as $\perp$.

Say there are some kind of basic RArs which hold only one dimension of resource need. For instance, one RAr, named A_1, may only need some CPU capacity for computation while another, named A_2, just need some memory capacity to record information. It is easy to verify that not all cluster devices which can service A_1 can service A_2 and vise versa. So in this case, A_1 and A_2 are parallel while $\perp \preceq A_1$,A_2. Then it is also easy to extend this instance to the real case that there will be a large number n of different dimensions of RArs, and we have the following two conditions:

$$1) \ \forall i, j \in n, A_i \parallel A_j$$

$$2) \ \forall i \in n, \perp \preceq A_i$$

Now we consider the union of those RArs A_i. Some more complexed RArs may contain multi-dimensional resource allocation requests, which can be viewed as request different one-dimensional RArs simultaneously and this can be achieve by the union of A_i. In this way, we can define a n-dimensional RAr as follows:

Definition 3 *Given A^n denote a n-dimensional RAr, then:*

$$A^n = (\forall A_i \ i \in n) \bigwedge A_i.$$

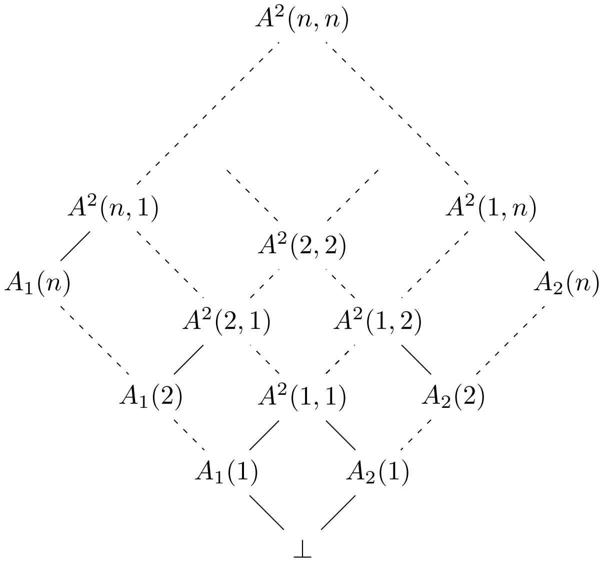

Figure 1: 2-dimensional(CPU, Memory) RAr topology

Finally, we say any kind of resource allocation request can be viewed as a n-dimensional RAr and the topology of RArs model can be built up as a partially order set through the relations we just defined. We give an instance of two-dimensional RAr topology in **Figure 1**. We use $A_1(1)$ to represent a one-dimensional RAr which only require 1GHz CPU and $A_2(1)$ to represent another one-dimensional RAr requiring 1GB memory. Then $A^2(1,1)$ is the union of them and represent a two-dimensional RAr. For easy illustration, we suppose 1GHz CPU and 1GB memory are two atom (or say entry) requests in this instance. As there may be 1.5GHz CPU request in a RAr, we use dash line to denote the relation which contains intern nodes and full line to denote the closed relation. It is not a supervise that we find the result topology is actually a modular lattice.

Note: Instead of the theoretical analysis as above, another way of generating the topology of resource allocation requests, or more generally say *informations*, is through real data experiment. One instance of this approach can be refer to [3] which tested the web-service requests and also resulted a partially order set.

5 MPC-X RArs

Now we consider the MPC-X system with the additional constraints between RArs as a practical case. This is a real extended scenario that a RAr can request different server to work for it or say allocate to it simultaneously which break the classical constraints for any bin-packing problem model as they all limit that one ball can only be allocated to exactly one bin.

In this section, we first briefly describe the topology of MPC-X device and then model the RArs based on its special relations.

5.1 MPC-X topology and RAr relations

An abstract MPC-X device contains eight servers arranged in a ring. We assume the servers are homogeneous in a device. Then there are two types of RArs: one for singleton servers whilst another for adjacent servers. We use S_n (n=1...8) to represent the first type RArs while A_n (n=1...8) to represent the second and in both of them, the index represent the number of server it requested.

Then it is easy to say the foundation relation below:

$$1)\ \forall i \in (1, ..., n-1), S_i \preceq S_{i+1}$$

$$2)\ \forall i \in (1, ..., n-1), A_i \preceq A_{i+1}$$

$$3)\ \forall i \in (1, ..., n), S_i \preceq A_i$$

$$4)\qquad A_1 = S_1$$

$$5)\qquad A_7 = S_7$$

$$6)\qquad A_8 = S_8$$

$$5)\qquad A_2 \preceq S_5$$

$$6)\qquad A_3 \preceq S_6$$

5.2 Special RAr model

We give the RArs model of the MPC-X device below **Figure 2**, ignored the totally-ordered part for easy illustration. As mentioned in section 4.1, we apply Birkhoff's representation theory to map this partial order set to a downset lattice which can be referred to **Figure 3**. We use B_i to label the graph instead of using the downset label directly to make it more concisely. To illustrate the correspondence, for example, we know B_1=S_2 while B_{14} is the downset of $S_6 \cup A_4$ which contains $\{S_6, S_5, S_4, S_3, S_2, A_4, A_3, A_2\}$.

Note: A general result we can inferred from this instance is that the RAr topology from any practical or say industrial distributed systems can be modelled as a modular-lattice by our representation and Birkhoff's theorem.

Let's then focus on the new modular-lattice model, actually it contains some really additional *interesting* nodes which we just ignored in our first MPC-X RAr topology. For instance, we still use the B_{14} we just mentioned above for

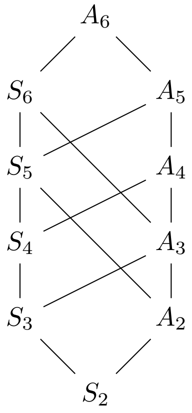

Figure 2: MPC-X RAr topology

illustration. There cannot be any single node in the pervious topology to exactly represent the scenario of the union of S_6 and A_4. (If we say A_6, then we find that its capacity is beyond our needs as a scenario in which six servers can work while five of them are adjacent can serve our need but it doesn't satisfy A_6.) On the other hand, the total sets of node in this new modular-lattice model can represent all the possible scenarios of RAr for the device based on its downward-close character.

5.3 Metric on RAr model

At this step, we can begin to consider a metric on this lattice-model to represent the distance between different RArs. A common metric on modular-lattice is a *valuation* computed by a ranking function.

6 Evaluation

All projects need to contain a serious and careful evaluation of their results. The specifics of the evaluation method (e.g., user study, experiments, formal proof review, etc.) are intrinsic to the nature of the project, so this is something that you must discuss and agree with your supervisor early in the project. Ideally, a presentation of the method and results of your evaluation should be included in its own separate section of the report.

6.1 Evaluation architecture

Three prototype programs are implemented in this step. The first one is a one-dimensional bin packing program by basic linear programming approach. It

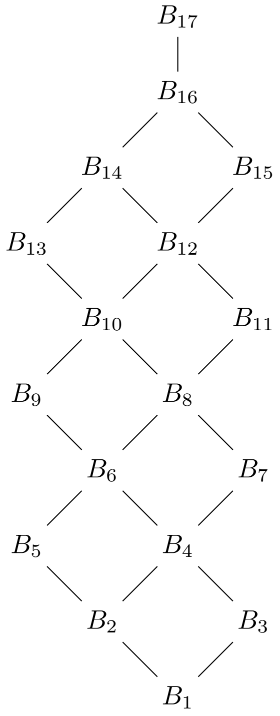

Figure 3: MPC-X RAr lattice-topology

contains a tiny input size composed by 3 balls and 2 bins and a basic greedy search algorithm which we used as the benchmark of our following experiment. The second one is a two-dimensional bin packing program which is trivially extended from the first one by parallel adding the second dimension. It contains a little larger input size compared with the first one which we used as the basic input instance for multi-dimensional pin packing program.

Our new model is implemented from the third prototype program which is a one-dimensional bin packing program with the same tiny input size of program one, but the allocation is represented by a lattice which using the domain of bins to represent the balls and the values in the domain to represent the size of each bin.

6.2 Evaluation Results

We compare the program one and three. When both of them find the same solution, the model building time and resolution time used for program three are all less than the program one.

7 Conclusions and Future work

All good projects conclude with an objective evaluation of the project's successes and failures, and suggestions for future work that can take the project further. It is important to understand that there is no such thing as a perfect project. Even the very best pieces of work have their limitations and you are expected to provide a proper critical appraisal of what you have done. Your assessors are bound to spot the limitations of your work and you are expected to be able to do the same.

Appendices

Appendices contain information that is peripheral to the main body of the report. Information typically included are things like program listings, tables, proofs, graphs, or any other material that would break up the theme of the text if it appeared in situ. Large program listings are rarely required, and should be compressed as much as possible, e.g., by printing in multiple columns and by using small font sizes, omitting inessential detail.

Literature Review

From [8]:

Industrial Background: using virtual machine (VM) technology which can consolidate hardware resources to enable shared hosting platforms.

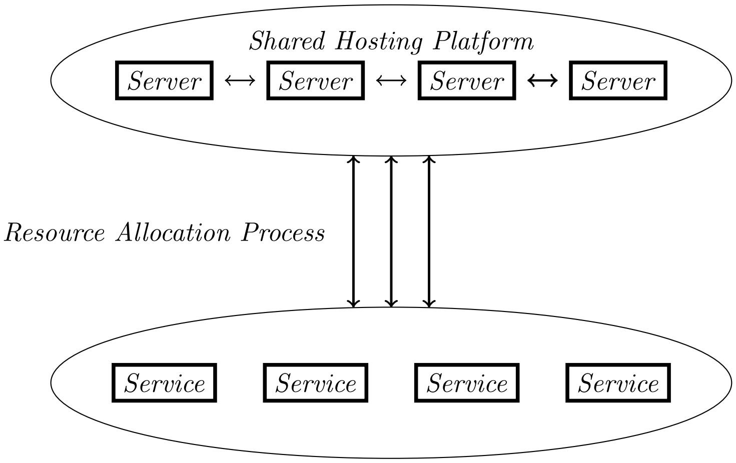

Figure 4: System Model

Main achievement: A formulation of the resource allocation problem in shared hosting platform for static workloads to make decisions when allocating hardware resources to service instances.

System model: Show in **Figure 4**

Resource and Service model: Each server provides several resources (e.g., CPU, RAM space, I/O bandwidth, disk space) and we consider two kinds of resource needs: *rigid* and *fluid*. A rigid need denotes that a specific fraction of a resource is required. The service cannot benefit from a larger fraction and cannot operate with a smaller fraction. A fluid need specifies the maximum fraction of a resource that the service could use if alone on the server. The service cannot benefit from a larger fraction, but can operate with a smaller fraction at the cost of reduced performance. For each fluid resource need we can define the ratio between the resource fraction allocated and the maximum resource fraction potentially used. We call this ratio the *yield* of the fluid resource need. To compare yields across services with various minimum yield requirements, we define the *scaled yield* of a service as follows:

$$\text{scaled yield} = \frac{\text{yield} - \text{minimum yield}}{1 - \text{minimum yield}} .$$

After that, we can specify the assumptions and list them below:
1) Each service consists of only a single VM instance.
2) Each service has constant resource needs.
3) The yields of all fluid resource needs are identical.
4) Rigid resource need are independent from fluid resource needs
5) User concerns, or say higher level metrics are directly related to resource

fractions allocated to services

Then we specify the aim precisely as *Maximize the Minimum Yield* under the two constraints below:
1) The resource capacities of the servers not be overcome.
2) A service inside a single VM instance should be allocated to a single server.

Finally, we recall the problem formulation as a Mixed Integer Linear Program (MILP). Say we have N services indexed by i, H servers indexed by h which each provides d types of resources. Fractions of these resources can be allocated to services. For each service i, r_{ij} denotes its resource need for resource type j, as a resource fraction between 0 and 1. δ_{ij} is a binary value that is 1 if r_{ij} is a rigid need, and 0 if r_{ij} is a fluid need. We use $\hat{y}_i$ to denote the minimum yield requirement of service i, a value between 0 and 1. We define a binary variable e_{ih} that is 1 if service i runs on server h and 0 otherwise. We denote by y_{ih} the unscaled yield of service i on server h, which must be equal to 0 if the service does not run on the server. With these definitions the constraints of our linear program are as follows, with Y denoting the minimum yield:

$$
\begin{array}{rc}
\forall i, h & e_{ih} \in \{0,1\}\ , \quad y_{ih} \in \mathbb{Q} \qquad (1) \\
\forall i & \sum_h e_{ih} = 1 \qquad (2) \\
\forall i, h & 0 \leq y_{ih} \leq e_{ih} \qquad (3) \\
\forall i & \sum_h y_{ih} \geq \hat{y}_i \qquad (4) \\
\forall h, j & \sum_i r_{ij}(y_{ih}(1-\delta_{ij}) + e_{ih}\delta_{ij}) \leq 1 \qquad (5) \\
\forall i & \sum_h y_{ih} \geq \hat{y}_i + Y(1-\hat{y}_i) \qquad (6)
\end{array}
$$

About Vector Packing Algorithms: Resource Allocation Problem is related to the multi-dimensional version of bin-packing, or say vector packing: our service may have fluid resource needs. Attempt to place d-dimensional vectors into (at most) H bins.

From [6]:

A novel approach (Wrasse solver) to resource allocation that permits the problem specification to envelope with ease. The key contribution is defining a specification language for Resource Allocation, which is expressive enough to encode a multitude of allocation problems without using any domain-specific abstractions.

Face the fact that: there may be conflicting constraints: a strategy that is optimised for performance may not necessarily meet fault-tolerance requirement.

From [9]:

Focus on the theoretical model: In addition of [8], as heterogeneous cases, we consider each server's each resource (or say node's dimension) has two different capacity: Elementary capacity: a single element in each dimension; Aggregate capacity: total resource capacity counting all elements.

Compared the MILP model with [8]: Using different vector to represent requirement and needs separately instead of invloving a binary indicator. Add one more constraint to represent the aggregate resource capacities achieved by sum the resource used from all services on this node.

Still consider the allocation under two constraints: Rigid requirement and fluid need; Both of them become ordered vector pair. Still precisely define the problem as maximize the minimum yield over all services.

From [4] and [1]:

Notes from Heuristics for Vector Bin Packing [4] and Lower bounds and algorithms for the 2-dimensional vector packing problem [1] –Focus on the new Heuristics and Experiments model, the way to set up the experiment environment.

FFD(First Fit Decreasing) Bin-centric view: Open only one bin at any time, place items into this bin from the largest suitable one. Close the bin when no item can be put in. Heuristics: Involve random choosing. Grasp[k]: pick a random one from the best k instead of the bext one. Bubblesearch: the kth best is chosen with a propobability proportional to $(1-p)^k$, for a suitable p.

Analysis of FFDAvgSum, which is same as VP_CPSum in[8]: For the case when some demands in a dimension always dominate the demands in the other dimension, FFDSum is more robust, the dimensions that are not scare can be assigned smaller coefficients and have a smaller impact on the ordering. Analysis [8]: the demands across the dimensions were simpled i.i.d.

Invloving classes of randomly generated instances to model the 2-dimensional case. Using two variables c,d to represent the bins capacities on different dimensions, respectively. Using Wj and Vj to represent the weights of item j on different dimensions, respectively. Using u.d.[a,b] to represent the uniformly distribution of the value for weights of item in the interval [a,b]

In [4] and [1], the dimensions in the first six classes are independently sampled. In classes seven and eight, the dimensions are correlated. They achieve positive correlation by setup the domain of the Vj to be a monotonic function from Wj while achieve negative correlation by setup it to be a inverse function in domain.

In [4]: Trivial way to extend to multi-dimension is to set dimension (2i-1) and (2i) are correlated as dimension 1 and 2, while independent of the other dimensions. As for generate negative correlation across all dimensions: Random variables to denote the random distribution of balls in each dimension, totally 2 times of the dimensions. Multiply with an random coeffieient from [10,40] and over two. Further noise by adding a random value from [0,1]. Ignored the overwight item.

From [11]:

This work presents an automatic virtual resource management system for Cloud infrastructures (hosting service platforms). The main advantages include: it can automate the dynamic provisioning and placement of VMs; support for heterogeneous applications and workloads; support for arbitrary application topology.

Focus on the automatic virtual resource management system:

It contains a more complicate system model, or say architecture, to simulate the resource allocation scenario which is more closed to the really world cloud infrastructures. Instead of using bins and balls to represent, this model composed by Application Environment (AE), Local Decision Module (LDM), Global Decision Module (GDM) and datacenter which contains physical machines and VMs. Briefly, we can say that AE is faced to the application associated with specific performance goals; An application-specific LDM is associated with each AE to evaluate the process baed on the current workload using service-level metric and generate a utility function of the resource allocation; A GDM is used to interact with each LDM and the real-datacenter. It is the core of this system which contains the Constraint solver to determine the management actions based on the input of LDM's utility functions and datacenter's system-level performance metrics. It works like a black box compared with LDM. We prefer to refer to the fig.2 in [11] of the system graph.

Two utility functions in LDM: a fixed service-level function and a dynamic resource-level function which is communicated with GDM and updated for every iteration. VM allocation vectors in LDM are used for build up the upper bound constraints given by each application.

As for the GDM: It involve two sequential process, one for determine VM allocation vectors for each application (VM Provisioning) and then for placing VMs and PMs and achieving optimisation (VM Packing). Beyond the constraints formulations based on the CPU and Memory, an interesting approach in VM Provisioning is using a coefficient to allow the administrator to trade-off between the fulfilment of the performance goals and the cost of operating the required resources. Another interesting method is illustrated during the optimisation process in VM Packing: To minimise the number of migration required to reach the new VM-to-PM assignment, or say minimise the reconfiguration to provide a strategy with few interim steps and maximum degree of parallelism.

From [10]:

Instead of pervious works, this paper model the resource allocation problem based on the job-scheduling scenario which put special emphasis on the working-time issue. It considers the randomness by involving Poisson processing during service time. Then it solves the new issues aroused by this model which including probability distribution and job blocking. It analysis the different cases for VMs, job sizes and release time, respectively.

From [7]:

This paper introduces a dedicated constraint using pruning and propagation rules with lower bound method to significantly reduce search on one dimensional bin packing problem.

Instead of the constraints it gives, we are actually more interested in his problem-model. Comparing with other works, it models the problem from a view of *set theory*: It mentions the goal is to assign each variable an element from its domain, then the solution can be viewed as an assignment. During the solution construction, we obtain a partial assignment which is a set of current domains at each step. Then it can views the optimisation process as a domain reduction on variables. And it clearly illustrates that its constraint includes one parameter which is a vector of n element (n is the number of balls) indicating the index of the bin into which it will be placed. This is very similar with our approach as modelling the relation of bins and balls by viewing one is the value in another's domain but from another direction.

The new constraint it obtains is by considering the feature of candidate set of each bin. Trivially, there is a bound that a ball whose size is larger than the bin capacity should not be viewed as in this bin's candidate set. It gives some more precisely mathematical notations to represent the relation and generate this condition to make it dynamically during the searching process. In detail, it gives the formulas of single item elimination and commitment. A important fact it mentions is that any bin load is equal to the total size to be packed, minus the loads of all other bins as it is the only rule which communicates information between different bins.

It also mentions that it is NP-complete to achieve generalised arc consistency in packing constraint, so instead it describes a not-complete algorithm by considering *neighbouring subsets* in candidate sets. It means two sets are neighbouring if there is no other subset in the candidate set whose items sum to a value strictly between themselves. It implement an linear time (even can be computed in constant time by dynamically updating) algorithm *NoSUM* to determine if there is such a interim subset exist. Then it is trivially to prune the problem instance by invoking this algorithm. The new lower bound it involved is by splitting the balls in candidate set into subsets using a constant number and apply the result to partial solutions. In conclusion, this new constraint cut search by orders of magnitude.

User Guide

For projects that result in a new piece of software, you should provide a proper user guide providing easily understood instructions on how to install and use it. A particularly useful approach is to treat the user guide as a walk through of a typical session, or set of sessions, that collectively display all the features of your software. Technical details of how the software works is rarely required. Keep it concise and simple. The extensive use of diagrams illustrating the software in action prove particularly helpful. The user guide is sometimes included as a chapter in the main body of the report, but is often better as an appendix to the main report.

We use Java-Choco [5] solver to implement the experimental programme.

References

[1] Alberto Caprara and Paolo Toth. Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, 111(3):231–262, 2001.

[2] Brian A Davey and Hilary A Priestley. *Introduction to lattices and order*. Cambridge university press, 2002.

[3] Heikki Mannila and Christopher Meek. Global partial orders from sequential data. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 161–168. ACM, 2000.

[4] Rina Panigrahy, Kunal Talwar, Lincoln Uyeda, and Udi Wieder. Heuristics for vector bin packing. *research. microsoft. com*, 2011.

[5] Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco3 Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2014.

[6] Anshul Rai, Ranjita Bhagwan, and Saikat Guha. Generalized resource allocation for the cloud. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 15. ACM, 2012.

[7] Paul Shaw. A constraint for bin packing. In *Principles and Practice of Constraint Programming–CP 2004*, pages 648–662. Springer, 2004.

[8] Mark Stillwell, David Schanzenbach, Frédéric Vivien, and Henri Casanova. Resource allocation algorithms for virtualized service hosting platforms. *Journal of Parallel and Distributed Computing*, 70(9):962–974, 2010.

[9] Mark Stillwell, Frédéric Vivien, and Henri Casanova. Dynamic fractional resource scheduling versus batch scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 23(3):521–529, 2012.

[10] Shahin Vakilinia, Mustafa Mehmet Ali, and Dongyu Qiu. Modeling of the resource allocation in cloud computing centers. *Computer Networks*, 91:453–470, 2015.

[11] Hien Nguyen Van, Frederic Dang Tran, and Jean-Marc Menaud. Sla-aware virtual resource management for cloud infrastructures. In *Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on*, volume 1, pages 357–362. IEEE, 2009.