Multi-objective optimization of data flows in a multi-cloud environment

Efthymia Tsamoura
Aristotle University of Thessaloniki, Greece
etsamour@csd.auth.gr

Anastasios Gounaris
Aristotle University of Thessaloniki, Greece
gounaria@csd.auth.gr

Kostas Tsichlas
Aristotle University of Thessaloniki, Greece
tsichlas@csd.auth.gr

ABSTRACT

As cloud-based solutions have become one of the main choices for intensive data analysis both for business decision making and scientific purposes, users face the problem of choosing among different cloud providers. In this work, we deal with data analysis flows that can be split in stages, and each stage can run on multiple cloud infrastructures. For each stage, a cloud provider may make a bid in the form of a continuous function in the time delay-monetary cost domain. The goal is to compute the optimal combination of bids according to how much a user is prepared to pay for the total time delay to execute the analysis task. The contributions of this work are (i) to provide a solution that can be computed in pseudo-polynomial time and with bounded relative error for the generic case; (ii) to provide exact polynomial solutions for specific cases; and (iii) to experimentally evaluate our proposal against other techniques. Our extensive results show that we can yield improvements up to an order of magnitude compared to existing heuristics.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems

1. INTRODUCTION

Data analysis tasks are typically represented as directed acyclic graphs (DAGs), which can be naturally split in multiple stages. In the past, data-driven analysis almost exclusively relied on database technologies, and the tasks were query execution plans consisting of operators from the extended relational algebra. Nowadays, data-driven decision support and scientific research involves more complex DAG flows that encompass data/text analytics, machine learning operations, and so on [6]. Additionally, cloud computing has emerged as a cost-effective solution to perform data analytics without requiring the investment on large-scale proprietary computing infrastructures; rather users pay only for the resources they use. Another attractive feature of cloud computing is elasticity, which allows for dynamic allocations of machines in order to cope with user and task requirements. This setting naturally gives rise to new problems in the optimization of data analysis processes, the objectives of which need to account for both performance and monetary cost.

Our setting is depicted in Fig. 1. We assume that our analysis process is a DAG that is split into stages, which will be referred to as *strides*. For each stride, each cloud provider may provide a bid, which includes the expenses involved and the profit. Because of elasticity and virtualization, each cloud host can offer multiple combinations of size and number of virtual machines at a different cost, and each such combination may result in different expected execution time. In the generic case, the complete offer per provider per stride is described by a continuous function. In addition, we assume that each user specifies her/his own function that represents the worst acceptable trade-off. We further assume that the total monetary cost of the data analysis process is the sum of the cost of each stride; similarly, the total time delay is the sum of the delays in each stride. We take a user-oriented approach, and our goal is to compute the combination of the bid points, so that there is exactly one bid point from each stride, and the total monetary cost maximizes the difference from the user-supplied function. We define *user satisfaction* as that difference (see Fig. 1), which captures the savings from the worst acceptable payment.

The problem above involves the computation of the pareto frontier and is NP-hard [5]. A simpler version of this problem has been investigated in the context of the Mariposa distributed query processing system [7, 5]. The major difference from our work is that, in Mariposa, the bid of each provider for each stride is a single point rather than a continuous function, implying that there is no provision of elasticity. Our contribution is that we provide solutions for the extended case described above. We begin with the assumption that the cloud host bids are piece-wise non-increasing

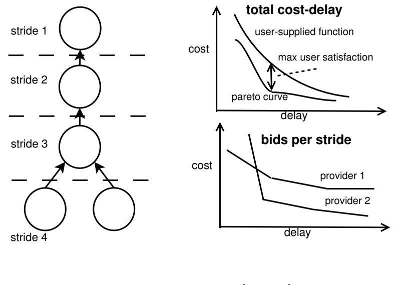

Figure 1: An analysis DAG (left) and example user and provider cost-delay functions (right).

linear functions, and we distinguish between convex and non-convex user-supplied functions. Finally, we generalize for the case, where the cloud provider functions are arbitrary non-increasing functions, and we also briefly discuss the case, where the functions are stepwise ones. Our theoretical results are that, in the generic case, we can find a solution with bounded relative error in pseudo-polynomial time, but we can also find exact solutions in a range of other more specific cases. Our evaluation results show that we can increase the user satisfaction by up to an order of magnitude.

The structure of the paper is as follows. We briefly review Mariposa in Sec. 2 and we formally state the problem in Sec. 3. Our solutions for convex and non-convex user-supplied trade-off functions are given in Sec. 4 and 5, respectively. The experimental analysis is in Sec. 6. We discuss the related work in Sec. 7 and we conclude in Sec. 8.

2. BACKGROUND

The Mariposa optimization algorithm and its extension in [5] form the basis of our work, on top of which we build our solutions. Mariposa's greedy algorithm aims at maximizing the user satisfaction in the case where each cloud provider makes a single bid per stride in the form of a point (c, d) in the 2-dimensional time delay-monetary cost space [7]. To avoid the exponential complexity of examining each combination of bids per stride, the algorithm first computes the bids that constitute the vertices of the pareto curve of each stride separately [5], which has a polynomial cost. In this way, no *dominated* bids are considered, i.e., bids for which there is at least one other bid with the same or smaller time delay and monetary cost. Then, those vertices are examined in order of their time delay. At the first step, the bids that minimize the time delay for each stride are chosen. In each iteration, one of the bids is replaced by its subsequent one in the relevant convex pareto curve; the choice is based on the gradient of the convex pareto curve.

In summary, the theoretical results regarding Mariposa's solution are firstly, that an optimal solution can be found in polynomial time under convex user-supplied trade-off functions (with the help of the greedy algorithm) and, secondly, the problem is NP-hard under non-convex user functions [5]. The former result implies that the optimal solution in that case lies on the convex pareto frontier rather than the complete pareto curve. The second result states that, if the user function is not convex, the optimal solution may lie anywhere on the pareto curve, which renders the problem intractable. In that case, the optimal point can be found in pseudopolynomial time with the help of dynamic programming; however it is shown that it can be approximated in polynomial time and the approximation has a bounded relative error.

3. PROBLEM FORMULATION

Our problem is formally stated as follows (see also Table 1 for the notation). A data analysis task is divided in N horizontal strides, and each one of the M cloud providers makes a bid in the form of a piecewise linear function of time delay d denoted as $l_{i,j}(d), 1 \leq i \leq N, 1 \leq j \leq M$; we will relax this restriction later. Each bid function contains $m_{i,j}$ linear segments. Also, the user provides a worst acceptable trade-off budget function $u(d)$. We want to maximize user satisfaction, which is formally stated as:

$$u(d^\star) - c^\star = \arg\max\{u(d) - c\}, \tag{1}$$

Table 1: Notation Table

Symbol	Description
N	Number of horizontal strides
M	Number of cloud providers
$u(d)$	Non-increasing user-supplied worst acceptable payment as a function of execution time d
$\mathbb{S}$	Set of feasible solutions, i.e., combinations of bids
$s = (d, c)$	A feasible solution of $\mathbb{S}$ with total time delay d and total monetary cost c
$\underline{d}$, $\overline{d}$,	Minimum and maximum delay of a feasible solution
$s^\star = (d^\star, c^\star)$	Optimal solution with total delay $d^\star$ and total cost $c^\star$
$\mathcal{S}$	Set of solutions returned by our solutions
P	Pareto curve of $\mathbb{S}$
CP	Convex Pareto curve of $\mathbb{S}$
P_i	Pareto curve of the bids in the i-th stride
CP_i	Convex Pareto curve of the bids in the i-th stride
$l_{i,j}(d)$	Non-increasing piecewise linear bid function of the j-th provider for the i-th stride
$m_{i,j}$	Number of linear segments of $l_{i,j}$
$(d_{i,j}^{k-1}, c_{i,j}^{k-1})$, $(d_{i,j}^{k}, c_{i,j}^{k})$	Endpoints of the k-th line segment of $l_{i,j}(d)$, $1 \leq k \leq m_{i,j}$
$\lambda_{i,j}^{k}$	The slope of the k-th line segment of $l_{i,j}(d)$
λ_{max}	Maximum slope of $l_{i,j}(d)$
$\mathcal{I}_i$	Intersection points of $l_{i,j}$ bid functions $\forall j$
$\mathcal{V}_i$	The set of endpoints that belong to CP_i
$v_{i,k}$	$k-th$ point in $\mathcal{V}_i$
ϵ	Pareto relative approximation error

In Eq.(1), $d^\star$ and $c^\star$ are the delay and the monetary cost of the optimal solution $s^\star$ of a flow, respectively. The delay d and the cost c of a solution $s = (d, c)$, in turn, are given by the sum of the delay[1] and cost values for each stride. The monetary cost and the time delay are expressed in cost and time units, respectively. Those can be real-world ones, such as dollars and minutes, or logical.

4. CONVEX USER FUNCTIONS

As stated in Sec. 2, when each provider submits a single-point bid per stride, the problem is tractable if the user trade-off function is linear or convex, and, in that case, the optimal solution lies on the convex Pareto curve CP of the global solution set $\mathbb{S}$. Moreover, the solution can be computed polynomially with the help of a greedy algorithm ([5]). In this section, we claim that we can compute the optimal solution in polynomial time, even when the providers submit continuous piecewise linear bid functions.

Our solution is termed as Continuous Bid-Convex User function (CBCU). It reduces the continuous bid functions to a set of single-point bids, and then it applies the greedy algorithm of [5]. More specifically, from the piecewise linear bid function for each stride, CBCU selects only the set $\mathcal{V}_i$, which comprises the endpoints of all segments and their intersection points that also belong to the convex pareto curve of the i-th stride, CP_i. In other words, the continuous bids are reduced to a set of single-point bids that are also vertices of $CP_i, \forall i$. The points in $\mathcal{V}_i$ are sorted in increasing delay, decreasing cost order.

THEOREM 1. *CBCU returns the optimal solution for linear or convex user-supplied functions u.*

PROOF. The proof proceeds in the following steps. First, we prove that the optimal solution that lies on CP can be generated by considering individually each CP_i. Then, we prove that although we do not examine the entire curve but only vertices, we do not miss the optimal solution. In the

[1]We ignore inter-stride communication costs, which is something we plan to investigate in the future.

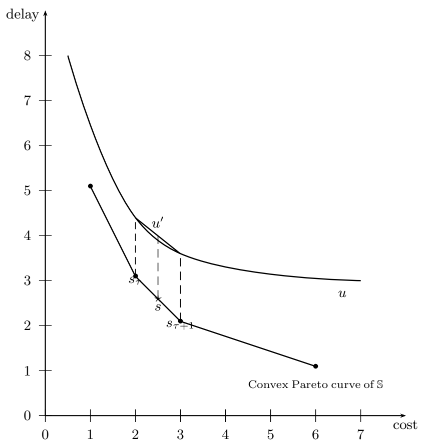

Figure 2: A schematic interpretation of Theorem 1.

following, let $\mathcal{S}$ be the set of solutions returned by the algorithm.

To begin with, we prove that a solution that belongs to CP consists solely of bids belonging to each CP_i, $i = 1, \ldots, N$. We know that any vertex of CP is the optimal solution of a scalar objective delay/cost function $w_1c + w_2d$, $w_1, w_2 \geq 0$. In our case, the optimization objectives are linear, and more specifically, the delay and cost of a solution are the sum of the delay and cost values of each constituting bid. Thus, we can optimize $w_1c + w_2d$ by optimizing each stride separately, i.e., by choosing a bid point $v_{i,j}$ from each stride that optimizes $w_1c + w_2d$. This also implies that each bid point $v_{i,j}$ of a solution s must belong to CP_i.

Next, we prove that CBCU can compute the complete CP. We know that the edges of any convex Pareto curve (each one of them is formed by taking two consecutive vertices ordered by increasing delay) are in decreasing slope order. This property also holds for the sequence of solutions $\mathcal{S}$ given that (i) the greedy algorithm visits the strides after ordering the corresponding edges $(\{v_{i,k}\}, \{v_{i,k+1}\})$ in decreasing slope and (ii) in every iteration τ, it forms the solution s_τ that results in the maximum slope edge $(s_{\tau-1}, s_\tau)$. This implies that $\mathcal{S}$ is the convex pareto curve of $\mathbb{S}$, since otherwise the maximum slope edge would not have been chosen by the algorithm.

The proof is completed by showing that the optimal solution belongs to $\mathcal{S}$, i.e., the set of solutions returned by CBCU when employed with input the sets $\mathcal{V}_i$ (endpoints). The latter is proved, in turn, if we show that the user satisfaction for any point s in CP is not larger than the maximum of the profit of two points $s_\tau = (c_\tau, d_\tau)$ and $s_{\tau+1} = (c_{\tau+1}, d_{\tau+1})$ that satisfy the following properties: s_τ and $s_{\tau+1}$ belong to $\mathcal{S}$ and s lies on the line segment that passes through s_τ and $s_{\tau+1}$ (see also Fig. 2). Let u' be the line segment that passes through the points $(d_\tau, u(d_\tau))$ and $(d_{\tau+1}, u(d_{\tau+1}))$. Firstly, the distance between u' and s is not smaller than the distance between u and s, because u is convex. Secondly, the distance between u' and s is not larger the maximum of the distances between points s_τ and $s_{\tau+1}$ and u': the maximum user satisfaction corresponds to point $s_{\tau+1}$ if the slope of the segment $(s_\tau, s_{\tau+1})$ is steeper than the slope of u', and to point s_τ, otherwise. This completes the proof. □

Corollary 1. *The complexity of CBCU is in the worst case $O(\sum_{i=1}^{N} \mathcal{V}_i)$.*

```
Inputs
    $l_{i,j}$: providers' bid functions per stride
    $u$: user trade-off function
Outputs
    $s$: $c^\star + N\epsilon'$ or $(1+\epsilon)c^\star + N\epsilon'$ approximate solution.

 1: {Start of bid sampling}
 2: Compute the Pareto curve $P_i$ of each stride $i$;
 3: In each stride $i$ sample along cost with fixed step $\epsilon'$;
 4: Let $c^{\epsilon'}_{i,1}$, $c^{\epsilon'}_{i,2}$, ..., $c^{\epsilon'}_{i,k}$ be the positions along the cost
    axis that are considered.
 5: for all $c^{\epsilon'}_{i,k}$ do
 6:     Compute the delay $d^{\epsilon'}_{i,k} = P_i^{-1}(c^{\epsilon'}_{i,k})$;
 7:     Form the bid $p^{\epsilon'}_{i,k} = (d^{\epsilon'}_{i,k}, c^{\epsilon'}_{i,k})$; {Employing the
        above steps we collect $C_i/\epsilon$ samples per stride,
        where $C_i$ is the maximum cost in stride $i$.}
 8: end for
 9: {End of bid sampling}
10: Employ the pseudopolynomial DP or the polynomial
    algorithm of [5]; {Let $\mathcal{S}$ be the returned solutions
    (each one of which belongs to the $c^\star + N\epsilon'$ or
    $(1+\epsilon)c^\star + N\epsilon'$ Pareto curve of $\mathbb{S}$.)}
11: return the solution $s \in \mathbb{S}$ that maximizes Eq.(1);
```

Figure 3: The steps of CBNU and DP algorithms.

Proof. In the worst case, the greedy algorithm may visit exhaustively each set $\mathcal{V}_i$. This may happen when, in every iteration, the algorithm chooses bids from different strides. Note that $\sum_{i=1}^{N} \mathcal{V}_i = O(\sum_{i=1}^{N} \sum_{j=1}^{M} m_{i,j})$ □

5. NON-CONVEX USER FUNCTIONS

In the case of non-convex budget functions, CBCU cannot be applied, because it considers only points on the convex pareto curve, which is suboptimal. The methods in [5] cannot be directly applied either, because, in our case, the bids are continuous (i.e., there is an infinite number of points to consider). Therefore, our contrinution is that we propose a sampling algorithm that serves as the preliminary step for computing approximate Pareto curves (and consequently, approximate solutions). It results in solutions with approximation errors $c^\star + N\epsilon'$ or $(1+\epsilon)c^\star + N\epsilon'$ when combined with the pseudopolynomial or the ϵ-approximation algorithm presented in [5], respectively. We will use the acronyms CBNU (from Continuous Bid Non-convex User function) and DP (from Dynamic Programming) to refer to the combination of the proposed sampling algorithm with the ϵ-approximation and the pseudopolynomial algorithms of [5], respectively. We further assume that that the maximum slope of all piecewise linear bid functions is λ_{max}.

The steps of the proposed algorithms are shown in Fig. 3. The Pareto curve P_i of each stride (line 2) is computed in two steps. First, we form the set $\mathcal{U}_i$ by selecting the non-dominated points from $\mathcal{I}_i$ and all the endpoints of that stride. $\mathcal{I}_i$ is the set of points where the piecewise linear functions $l_{i,j}$ intersect. The points in $\mathcal{U}_i$ are then sorted in increasing delay order. Second, we check if any such pair $(u_{i,k}, u_{i,k+1})$, where $u_{i,k}, u_{i,k+1} \in \mathcal{U}_i$, is connected through (a part of) any $l_{i,j}$. In that case, the whole set of points lying on $(u_{i,k}, u_{i,k+1})$ belongs to P_i. During this process, the size of the set Δ of the sampled points is equal to $\sum_{i=1}^{N} C_i\epsilon'$, which is pseudopolynomial.

We prove the validity and the complexity of CBNU and DP through a sequence of lemmas. Our primary concern is the error induced by this sampling process, as well as how this process affects the time and space complexity of the algorithm.

LEMMA 1. *The error with respect to cost induced by the sampling process of the pareto fronts of the strides is additive and equal to $N\epsilon'/2$, where ϵ' is the sampling step and N is the number of strides.*

PROOF. Since the sampling step is ϵ', this means that the choice for the optimal solution for $\mathcal{S}$ with respect to cost for each stride is at most $\epsilon'/2$ away from the optimal solution with respect to the continuous space. As a result, we get a total of $N\epsilon'/2$ additive error. $\square$

This practically means that the cost of a solution returned by DP with respect to the optimal is $c^\star + N\epsilon'$. CBNU requires the existence of a delay unit. To ensure this, we make use of the maximum slope λ_{max} for the piecewise linear functions per stride.

LEMMA 2. *The delay unit for Δ is $\frac{\epsilon'}{\lambda_{max}}$.*

PROOF. Since the maximum slope is λ_{max} we get that for cost change equal to ϵ' the minimum delay change is equal to $\frac{\epsilon'}{\lambda_{max}}$ because of the linearity of the bid functions. $\square$

Based on these observations we can now state the main theorem of this section.

THEOREM 2. *The CBNU algorithm can compute an approximate solution with cost $(1+\epsilon)c^\star + N\epsilon'$ while the time complexity is $O(\frac{\lambda_{max}N|\Delta|}{\epsilon\epsilon'})$*

PROOF. The error bound is directly derived by Lemma 1 and the approximate algorithm in [5]. The time complexity is also derived from the complexity of the approximation scheme in [5] and the fact that the number of generated points in Δ is $\sum_{i=1}^{N} C_i\epsilon'$. $\square$

Generic functions. The above discussion can be extended also to arbitrary continuous functions for bids by the cloud provides. The only change is that λ_{max} is the maximum gradient of these functions; the gradient is given by the first derivative.

Stepwise functions. When the bids of the cloud providers conform to a stepwise function, then, the leftmost endpoint of each step segment dominates all the other points of that segment, because it corresponds to the same delay with lower cost. Consequently, we can discard all the other points, and the problem is reduced to a problem where all bids per stride are points. As such, the solutions in [5] apply.

Furthermore, if u is stepwise, the following theorem holds, according to which it is adequate to build the minimum cost plans with total delay equal to the delay of the rightmost endpoints of the step segments in u.

THEOREM 3. *If the user budget function u is stepwise, then we have to search for the optimal solution only at the rightmost endpoints of the step segments.*

PROOF. Let $(d, u(d))$ be the rightmost point of a step segment and $(d', u(d') = u(d))$, $d' \leq d$, be another point lying on the same step. Since for two plans s and s' of the Pareto curve with total delay d and d', respectively, the cost of s is not higher than the cost of s', then the distance between the cost of plan s and $u(d)$ is not lower than the distance between the cost of plan s' and $u(d)$. $\square$

Max delay \ Alg.	DP	CBNU$_{0.05}$	CBNU$_{0.5}$	G$-$MPT	G$-$MPM	SA$-$MPT
$N = 2$						
500	1	0.99	0.78	0.97	0.53	0.99
1000	1	0.98	0.61	0.96	0.45	0.99
10000	1	0.97	0.96	0.90	0.60	0.97
100000	1	0.92	0.90	0.70	0.90	0.91
$N = 5$						
500	1	0.96	0.82	0.84	0.70	0.70
1000	1	0.91	0.80	0.66	0.52	0.72
10000	1	0.84	0.75	0.28	0.67	0.66
100000	1	0.88	0.97	0.24	0.34	0.52
$N = 10$						
500	1	0.88	0.99	0.20	0.67	0.74
1000	1	0.87	0.53	0.17	0.80	0.29
10000	1	0.82	0.60	0.07	0.40	0.42
100000	1	0.83	0.70	0.05	0.20	0.30
$N = 20$						
500	1	0.95	0.21	0.14	0.23	0.23
1000	1	0.85	0.67	0.27	0.21	0.21
10000	1	0.84	0.77	0.09	0.12	0.06
100000	1	0.83	0.63	0.22	0.18	0.12
$N = 50$						
500	1	0.86	0.86	0.09	0.06	0.10
1000	1	0.91	0.54	0.42	0.46	0.38
10000	1	0.90	0.70	0.18	0.20	0.37
100000	1	0.92	0.49	0.18	0.23	0.30

Table 2: Normalized mean user satisfaction.

6. EVALUATION

In this section, we experimentally explore the performance and efficiency of CBNU and DP for non-convex budget functions; for convex budget functions CBCU is always optimal and particularly inexpensive. Performance relates to the magnitude of user satisfaction, while efficiency is evaluated by observing the running time spent to reach a solution. We compare both CBNU and DP with state-of-the-art algorithms that have been proposed in the literature for other multi-objective scheduling problems; these are G$-$MPT, G$-$MPM and SA$-$MPT [3]. In contrast to our work, their rationale is to employ single-criterion heuristics. G$-$MPT and G$-$MPM return the minimum delay and minimum cost solution, respectively. SA$-$MPT performs bid selection employing the simulated annealing meta-heuristic.

The experimental setting is as follows. We consider data flows with $N = \{2, 5, 10, 20, 50\}$ strides. The maximum delay of each $l_{i,j}$ may be up to 0.5K, 1K, 10K or 100K time units, while the maximum cost is up to 0.3K, 0.4K, 0.6K, 1K, 5K or 50K cost units. To allow a fair comparison of the techniques, the bid line segments do not dominate each other regardless of whether they are provided by one or multiple cloud providers. The length of each line segment is randomly selected from the following intervals (in time units): $[10, 60]$, $[10, 100]$, $[10, 200]$, $[10, 500]$ and $[10, 1000]$ (and the maximum slope λ_{max} 0.02, 0.04, 0.08, 0.14 or 0.20, respectively). I.e., for each possible assignment of N, we produce 120 test cases. Regarding the user function u, its maximum delay (cost) is N times the maximum delay (cost) of $l_{i,j}$. The maximum slope of each piece of u is $\{0.20, 0.45, 0.70, 1.4, 3.4, 6.8\}$. Overall, for each value of N we examine 720 combinations of user and bid functions. In order to evaluate the impact of the approximation factor ϵ on the running time and performance of CBNU, we conduct experiments with $\epsilon = 0.05$ (CBNU$_{0.05}$) and $\epsilon = 0.5$ (CBNU$_{0.5}$). The sampling step ϵ' for DP and CBNU is fixed to $\epsilon' = 1$ unit. The experiments are performed on a 40 core Linux machine with 1TB memory, but each algorithm runs as a single thread.

Tables 2 and 3 show the impact of the total number of strides and the maximum delay on the performance and running time of DP, CBNU$_{0.05}$, CBNU$_{0.5}$, G$-$MPT, G$-$MPM, SA$-$MPT algorithms, respectively. In both tables the values are normalized according to the behaviour of DP, which

Max delay \ Alg.	DP	$\mathrm{CBNU}_{0.05}$	$\mathrm{CBNU}_{0.5}$	G−MPT	G−MPM	SA−MPT
$N = 2$						
500	1	12.500	0.990	10^{-8}	10^{-8}	6.250
1000	1	8.333	0.800	10^{-8}	10^{-8}	3.125
10000	1	5.263	0.625	10^{-8}	10^{-8}	0.338
100000	1	4.167	0.572	10^{-8}	10^{-8}	0.040
$N = 5$						
500	1	4.297	0.538	10^{-8}	10^{-8}	0.046
1000	1	4.762	0.220	10^{-8}	10^{-8}	0.800
10000	1	1.639	0.152	10^{-8}	10^{-8}	0.103
100000	1	1.010	0.107	10^{-8}	10^{-8}	0.016
$N = 10$						
500	1	8.333	0.147	10^{-8}	10^{-8}	0.633
1000	1	4.348	0.112	10^{-8}	10^{-8}	0.356
10000	1	0.935	0.067	10^{-8}	10^{-8}	0.045
100000	1	0.513	0.055	10^{-8}	10^{-8}	0.012
$N = 20$						
500	1	10.000	0.107	10^{-8}	10^{-8}	0.337
1000	1	4.762	0.069	10^{-8}	10^{-8}	0.181
10000	1	0.606	0.032	10^{-8}	10^{-8}	0.022
100000	1	0.493	0.028	10^{-8}	10^{-8}	0.016
$N = 50$						
500	1	12.500	0.093	10^{-8}	10^{-8}	0.132
1000	1	1.333	0.052	10^{-8}	10^{-8}	0.022
10000	1	1.449	0.017	10^{-8}	10^{-8}	0.017
100000	1	1.786	0.014	10^{-8}	10^{-8}	0.009

Table 3: Normalized mean running times.

has the minimum approximation error. Table 2 shows that for flows with a few strides, we can build efficient solutions with heuristic single-criterion algorithms. If additionally, the maximum delay is small, SA−MPT's performance degradation is up to 3% only. However, DP can yield an order of magnitude increase in the user satisfaction compared to any heuristic solution, if the number of strides increases. Compared to $\mathrm{CBNU}_{0.5}$, the increase is up to 5 times, whereas $\mathrm{CBNU}_{0.05}$ leads to satisfaction degradation between 1% and 18%.

Regarding the running times (see Table 3), both G−MPT and G−MPM are several orders of magnitude faster than DP, since they simply select the minimum delay and minimum cost bids from each stride. Also, $\mathrm{CBNU}_{0.05}$ may have higher running time than DP in practice. This is due to the bid scaling step that is within CBNU, as explained in [5], which incurs significant overhead. Finally, $\mathrm{CBNU}_{0.5}$ and SA−MPT have approximately two orders of magnitude lower running times than DP for large numbers of strides. In absolute times, DP runs in the order of milliseconds for $N = 2$, up to more than an hour for $N = 50$. This time depends also on the maximum delay; if it is up to 1000 time units, which seems to be a realistic option, DP's running time is a few dozens of seconds even for 20 strides, and a few minutes for large flows of 50 strides.

In summary, the experiments above indicate that heuristic solutions can reach efficient solutions only for very small flows. DP outperforms $\mathrm{CBNU}_{0.05}$ both in terms of performance and running time. Finally, for large flows and large maximum delays, where DP's optimization overhead is significant, $\mathrm{CBNU}_{0.5}$ provides a good trade-off between performance and running time.

7. RELATED WORK

Kllapi et al. have studied the problem of cloud resource allocation under response time and monetary cost tradeoffs and have proposed the G−MPT, G−MPM and SA−MPT algorithms [3]. The two main differences from our work is that, first, their proposed heuristic algorithm performs single-criterion optimization, in contrast to our algorithms which optimize accounting for both criteria. Second, the resource pricing scheme differs to ours; although in [3] the resources have different processing/storage capabilities, their usage is charged uniformly, that is, each resource is charged based on the time it is used, while the monetary cost that must be paid for a specific time period is the same for all resources. As such, the solutions cannot extend to a multi-cloud setting.

Our work also relates to the area of utility-driven resource allocation. Depending on the utility functions, the objective may be to maximize the profit (e.g., [4], [1]) or the user satisfaction (e.g., [2]). Similarly to [3], the major difference from our work is that there are no multiple resource providers and both the monetary cost and the amount of allocated resources are a linear function of the time period during which the resources are occupied.

8. CONCLUSIONS

In this work, we provide solutions to the problem of allocating strides of data flows to cloud providers in order to maximize user satisfaction assuming that each provider makes a piece-wise linear continuous bid in the time delay - cost space. To this end, we propose algorithms that either compute the optimal trade-off, or provide an approximate solution with bounded relative error. Our experiments show that we can increase user satisfaction by up to an order of magnitude compared to existing heuristic solutions. Note, that our solutions also apply to the problem of profit maximization, if we assume that a cloud broker acts on behalf of a federation of providers and the user input describes the final amount to be paid by the user for different time delays. A main direction for future work is to consider the cost of shipping intermediate results across strides.

Acknowledgments. This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge society through the European Social Fund.

9. REFERENCES

[1] G. Feng, S. Garg, R. Buyya, and W. Li. Revenue maximization using adaptive resource provisioning in cloud computing environments. In *GRID*, pages 192–200, 2012.

[2] J. P. Hansen, S. Ghosh, R. Rajkumar, and J. P. Lehoczky. Resource management of highly configurable tasks. In *IPDPS*, 2004.

[3] H. Kllapi, E. Sitaridi, M. M. Tsangaris, and Y. Ioannidis. Schedule optimization for data processing flows on the cloud. In *SIGMOD*, pages 289–300, 2011.

[4] Y. C. Lee, C. Wang, A. Y. Zomaya, and B. B. Zhou. Profit-driven scheduling for cloud services with data access awareness. *Journal of Parallel and Distributed Computing*, 72(4):591–602, 2012.

[5] C. H. Papadimitriou and M. Yannakakis. Multiobjective query optimization. In *PODS*, pages 52–59, 2001.

[6] A. Simitsis, K. Wilkinson, M. Castellanos, and U. Dayal. Optimizing analytic data flows for multiple execution engines. In *SIGMOD*, pages 829–840, 2012.

[7] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: a wide-area distributed database system. *The VLDB Journal*, 5(1), 1996.