Comparison of Workflow Scheduling Using Constraint Programming or Auctions

Research Report 13-01-RR

by

Ferran Torrent-Fontbona
and
Beatriz López

Institute of Informatics and Applications
University of Girona
March 2013

Abstract

Business processes designers take into account the resources that the processes would need, but, due to the variable cost of certain parameters (like energy) or other circumstances, this scheduling must be done when business process enactment. In this report we formalize the energy aware resource cost, including time and usage dependent rates. We also present a constraint programming approach and an auction-based approach to solve the mentioned problem including a comparison of them and a comparison of the proposed algorithms for solving them.

Contents

List of Figures

Chapter 1

Introduction

Lately some energy supply companies and public institutions are making important efforts to start building the future smart grid. One of these efforts is delivering thousands of smart meters that would allow companies and customers measure their instant energy usage and it would lead to the application different time-dependent energy rates like the Spanish production market rate showed in Figure 1.1. Also, this smart meters would lead to the application of other strategies to smooth the daily energy usage curve or to make it more predictable, for example setting reduced prices for those organizations that keep their energy usage within some bounds.

Moreover, the improvement of energy management to reduce costs and CO_2 emissions has a social impact and also improves the stock market perfomance of the organizations [1], though most studies has been focused on house hold management [5]. To improve the energy management in organizations we propose the optimization of resource allocation in business process in order to minimize the costs associated to the energy usage.

This paper proposes a constraint programming method and an auction based method to perform the scheduling of workflows considering a time-dependent (time-dependent rate means that the per-resource-unit cost depends on when the resource is used) and usage-dependent rate for resources (usage-dependent rate means that the per-resource-unit cost depends on the

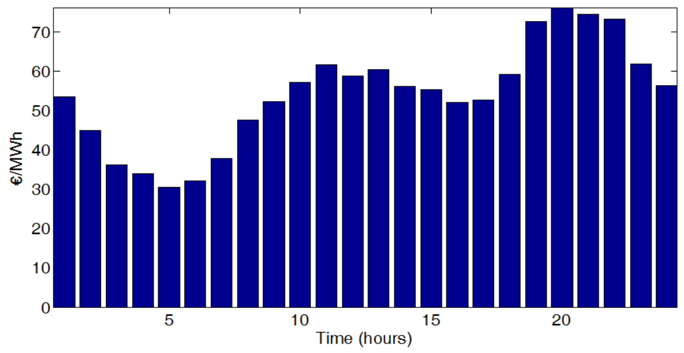

Figure 1.1: Average day hourly energy price of the Spanish production market in December 2012 according to [3].

load of the resource being used)[1][2]. We explore heuristic algorithms like Genetic Algorithms (GA) and Simulated Annealing (SA) to deal with some steps of the auction method because their complexity (NP-hard) due to the time-dependent and usage-dependent rates. We present an analysis of the elapsed time and the optimality of the solutions found by each method.

1.1 Related Work

There are few studies focused on the business management considering time dependent costs, but in [7] authors propose mixed integer programming and linear programming formulations solving them by Choco solver. Another work that considers time dependent costs is [8]. The main contribution of the paper is a new formulation for workflow scheduling considering time dependent costs and XOR nodes inside the workflows. Another contribution of the paper is that they propose solving the scheduling problem using combinatorial auctions and they propose a new compact formulation for bids, that we have used. Nevertheless this two works only consider time-dependent rates and not usage dependent rates.

Although energy variable costs has not been studied in depth, resource

[1]In electric terms load is related to the energy consumption

[2]usage-dependent rate is different from usage-mode used in [8]

allocation does. Many studies propose the use of auction mechanism to perform resource allocation. For example [11] presents a multi-attribute auction system for business processes scheduling, [2, 12, 4] presents different algorithms and mechanisms for deciding the winner for combinatorial auctions. We have used some mechanisms presented in these works for the Branch&Bound (B&B) algorithm like the tasks ordering, but we use a different bid formulation what avoids we cannot use most of the proposed mechanisms.

Chapter 2

Problem Definition

In this paper we are dealing with business processes (henceforth workflows) that consist of series of tasks which have to be done following a predefined order, therefore a workflow is an acyclic graph where each tasks can start its execution once all of its predecessor tasks have finished. Figure 2.1 shows a workflow example where nodes ST and ET indicate the earliest start time and lastest end time of the workflow respectively and where task 1 is the first task to be executed. Once it is completed, tasks 2 and 3 can start, and task 4 can start its execution once task 2 is finished.

Formally we define ST and ET as the window start time and end time respectively. So, $ET - ST$ is the maximum execution time of the workflow. We also define the set of tasks $\mathbf{T} = \{T_1 \ldots T_{\|\mathbf{T}\|}\}$. Each task $T_i = \langle [\underline{s_i}, \overline{s_i}], [\underline{et_i}, \overline{et_i}], RQ_i, \mathbf{PR} \rangle$ has an earliest and a latest start time, $\underline{s_i}$, $\overline{s_i}$, respectively, and an earliest and latest end time, $\underline{et_i}$, $\overline{et_i}$, respectively. RQ_i are the type of resources required by the task and $\mathbf{PR}$ is the matrix which

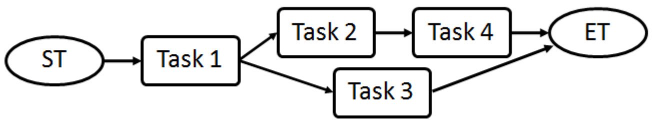

Figure 2.1: Workflow example. It illustrates that tasks 2 and 3 have to start after task 1 is done and task 4 have to start after task 2 is finished.

indicates which are the predecessor tasks or the tasks that must be executed simultaneously (if there are any). Each task has an associated duration and an energy usage, but they depend on the resource used. We define the set of available resources $\mathbf{R} = \left\{R_1 \dots R_{\|\mathbf{R}\|}\right\}$. Each resource R_k has an associated energy usage $e_{i,k}$, a duration $d_{i,k}$ and an additional monetary cost $m_{i,k}$ for deploying task i.

On the other hand, we are dealing with complex energy contract rates which include time-dependent prices but also reduced rates for those organizations that keep their energy usage within bounds, so they follow an agreed energy profile. The energy profile Σ is characterized by

$$\left\langle \underline{P_t}, \overline{P_t}, \underline{p_t}, \overline{p_t}, \underline{c_t}, c_t, \overline{c_t}, \underline{f_t}, \overline{f_t} \right\rangle$$

We define $\underline{P_t}$ and $\overline{P_t}$ as the minimum and maximum allowed energy usage at time t respectively, $\underline{p_t}$ and $\overline{p_t}$ as the minimum and maximum agreed energy usage at time t for reduced rate c_t (per-energy-unit cost). Then we also define $\underline{c_t}$ and $\overline{c_t}$ as the per-energy-costs when $p_t < \underline{p_t}$ or $p_t > \overline{p_t}$ respectively. We also consider the possibility that the contracted rate includes extra costs $\underline{f_t}$ and $\overline{f_t}$ (note that these are not per-energy-unit costs).

Figures 2.2 and 2.3 show two examples of energy profiles. First Figure 2.2 shows 5 zones depending on the energy usage that consists of two not allowed zones (for example for physical constraints), an agreed zone (where to keep the energy usage) which has a reduced rate and two non-agreed zones that if the energy usage belongs to them, the electric company will apply augmented rates, $\overline{c_t}$ and $\underline{c_t}$, or even fees, $\underline{f_t}$ and $\overline{f_t}$. The second example shows a 24 time slots rate[1] where the energy usage have to be between 0 and 250 energy-units in each time slot and the reduced rate zone depends on the time slot.

Then there are different costs associated to a scheduling: the makespan C_T is the time needed to perform the scheduling, the energy usage cost C_E is the energy-associated cost and it is related to the energy profile Σ, and the

[1]Smart meters would be able to measure the energy usage every 15 minutes, allowing companies to use rates with up to 96 time slots per day.

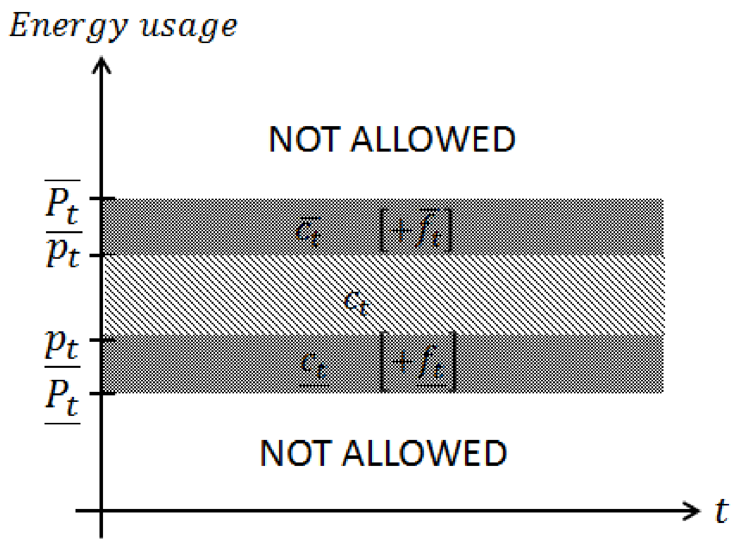

Figure 2.2: Energy profile example. There are unallowed zones (usually) due to physical constraints, and 3 different rates depending on the energy usage.

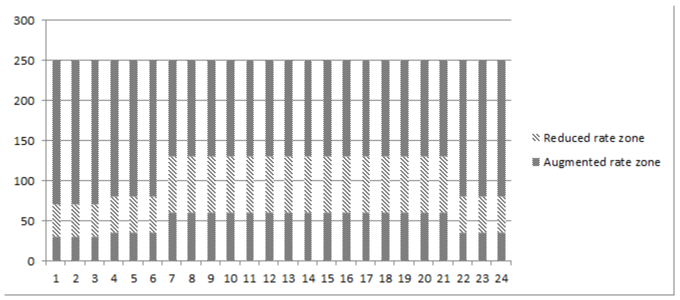

Figure 2.3: Energy profile example. Here is an example with 24 time slots where have to $0 < p_t < 250$ and the agreed energy usage depends on time.

resource usage cost C_R is the cost associated to other resources like machines or human resources. These costs will depend on the start time of each task s_i and the resources used for the execution of them.

The problem consists in given $\mathbf{T}$, $\mathbf{R}$ and Σ finding the start time s_i for each task and the resources needed to perform them that optimize the cost.

Chapter 3

Workflow Scheduling as Constraint Programming

In this chapter we present a MIP formulation in order to solve the workflow scheduling by a MIP solver.

For doing so, we followed [7], which propose to partition the entire resource area of the problem $[\underline{P_t}, \overline{P_t}] \times [ST, ET]$ (see Figure 3.1) with a collection of (sub)areas that do not overlap. Moreover we added the availability of lower bound energy usages and higher rates when are not respected (see Figure 3.1). Thus, the formulation presented in Chapter 2 is followed but we change the sub-indexes of those parameters related to these areas for convenience. Each area is defined inside a time interval; more than one area can occupy the same time interval but without overlapping one another according to Figure 3.1. Those areas within the same period of time define different rates according to the energy usage but they must be ordered by non-decreasing cost $y_i < y_j \Rightarrow c_i < c_j$. Similarly, but without any other constraint, we can state that those areas within the same energy usage interval define a time-variable rate.

Finally we define M as the number of defined areas. Each area A_j has a fixed initial position x_j (time dimension) and y_j (energy usage dimension), width w_j and height h_j and a per-energy-unit rate c_j. We also define a_j as the energy usage inside the jth area (the shadow area in Figure 3.1) and

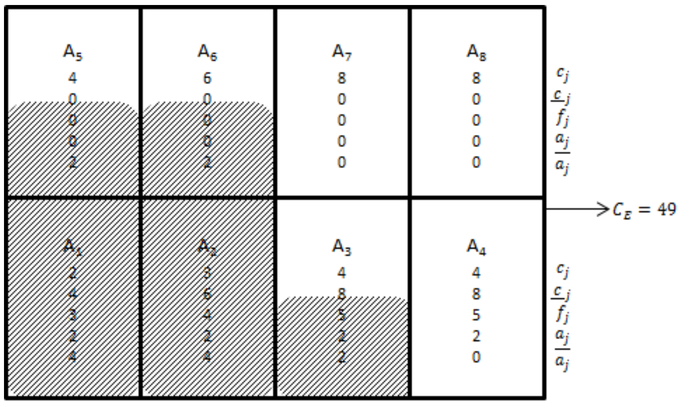

Figure 3.1: Resource area of the problem

a minimum agreed usage bound $\underline{a_j}$. Moreover each area has another per-energy-unit rate $\underline{c_j}$ and an extra cost $\underline{f_j}$ when the energy usage a_j is lower than the agreed lower bound $\underline{a_j}$.

Given the needed formulation we state the constraints of the problem:

$$\forall_i \; 1 \le i \le \|\mathbf{T}\| : \qquad ST \le \underline{s_i} \le s_i < s_i + \sum_{k=1}^{\|\mathbf{R}\|} z_{i,k} d_{i,k} \le \overline{s_i} + \sum_{k=1}^{\|\mathbf{R}\|} z_{i,k} d_{i,k} < ET \tag{3.1}$$

$$\forall_u \in Pr\,(i) : \qquad s_u + \sum_{k} z_{u,k} d_{u,k} < s_i \tag{3.2}$$

$$\forall_u \in Su\,(i) : \qquad s_i + \sum_{k} z_{i,k} d_{i,k} < s_u \tag{3.3}$$

$$\forall_u \in Sm\,(i) : \qquad s_i = s_u \tag{3.4}$$

$$\forall_t \; ST \le t < ET : \qquad \underline{P_t} \le p_t = \sum_{\forall_i | s_i \le t < si + \sum_k z_{i,k} d_{i,k}} \; \sum_{k=1}^{\|\mathbf{R}\|} z_{i,k} e_{i,k} \le \overline{P_t} \tag{3.5}$$

$$\forall_t \; ST \leq t < ET : \qquad \sum_{\forall_i | s_i \leq t < si + \sum_k z_{i,k} d_{i,k}} \sum_{k=1}^{\|\mathbf{R}\|} z_{i,k} \leq 1 \tag{3.6}$$

$$\forall_i \; 1 \leq i \leq \|\mathbf{T}\| : \qquad \sum_{k=1}^{\|\mathbf{R}\|} z_{i,k} = 1 \tag{3.7}$$

Where $z_{i,k}$ is a binary variable that indicates whether task i is assigned to resource k.

First we set the time constraints given by the time windows of the workflow and the corresponding task, (3.1) where each task cannot be scheduled before its earlier start time and has to be completed before its latest end time. Equations (3.2), (3.3) and (3.4) formalize the constraints given by the precedence constraints where a task i can not be executed before all its predecessor tasks, $Pr\left(i\right)$, have finished and it has to finish before its successor tasks, $Sr\left(i\right)$, start. Furthermore task i has to start at the same time as its simultaneous tasks $Sm\left(i\right)$. Equation (3.5) checks that the current energy usage p_t fulfills the energy profile. Equation (3.6) ensures that any resource is used over its capacity. Note that we can substitute 1 by another number allowing capacities higher than 1, but for simplicity we set the capacity of each resource to 1. Finally, equation (3.7) ensures that every task has assigned a single resource for its execution. [1]

Following we define the energy cost based on the energy cost at each time,

$$ov\left(t, p_t, A_j\right) = \begin{cases} \max\left(0, \min\left(y_j + h_j, p_t\right) - y_j\right) & x_j \leq t < x_j + w_j \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

[1]Note that we are stating that each task has to be executed by a single resource, however, tasks that need several resources at the same time can be easily considered using equation (3.4). For example if task T_i needs resource 1 and 2, we can divide it into T_{i1} and T_{i2} where both have the same start time, $s_{i1} = s_{i2}$.

$$\forall_j \; 1 \leq j \leq M : \qquad a_j = \sum_{\forall_t | ST \leq t < ET} ov\left(t, p_t, A_j\right) \tag{3.9}$$

Equation (3.8) defines the intersection between the energy usage p_t and the area A_j at time t. The global intersection between p_t and the area A_j is called a_j as equation (3.9) shows.

As said in Chapter 2, we consider three different costs: the resources cost, the energy cost and the make span. We define the energy cost of a schedule as the sum of the energy cost of each area and it depends on the contracted rate and the energy usage:

$$C_E = \begin{cases} \sum_{j=1}^{M} a_j c_j & \underline{a_j} \leq a_j \\ \sum_{j=1}^{M} a_j \underline{c_j} + \underline{f_j} & \underline{a_j} > a_j \end{cases} \tag{3.10}$$

Similarly, we define the resources monetary cost as:

$$C_R = \sum_{i=1}^{\|\mathbf{T}\|} \sum_{k=1}^{\|\mathbf{R}\|} z_{i,k} m_{i,k} \tag{3.11}$$

Finally, we define the makespan as the maximum end time less the minimum start time or, in other words, the maximum difference between a start time and an end time:

$$\forall_{i,j} \; 1 \leq i, j \leq \|\mathbf{T}\| : \qquad C_T = \max_{i,j} \left(s_i + \sum_{k=1}^{\|\mathbf{R}\|} z_{i,k} d_{i,k} - s_j \right) \tag{3.12}$$

Note that the variables of the formulated problem are s_i and $z_{i,k}$ $\forall_{i,k}$ since they define the schedule and the resources used for the execution of the tasks.

The problem is defined as the minimization of C_E or C_R or C_T.

According to that formulation, the Choco solver is able to find the solution.

Chapter 4

Workflow Scheduling by Auctions

Resource allocation problem has been widely studied and solved by a large variety of methods depending on the problem context. Sometimes, there are involved different agents or organizations in the resource allocation process which are independent and do not desire to share all their information with the others. Considering that context, the use of reverse auctions to perform the resource allocation is an appropriate method, [11, 8, 15, 2], since there is an organization (auctioneer) that demand a workflow, with a set of tasks, to be performed by resource agents that compete to perform the tasks they are enable to, to increase their utility.

Thus we developed an auction simulator to perform resource allocation considering that there is only one auctioneer and that each resource agent manages a set of resources that able it to send bids to perform the auctioned tasks. Then the auctioneer solves the winner determination problem to set which tasks perform each agent and when.

4.1 Bidding strategy

The auctioneer sends the requests for proposals (a proposal per task) to the resource agents. Therefore, each agent can have several tasks $T_i \dots T_j$ to consider which led the agent to explore a considerable amount of combinations. Therefore, the price of each bid could depend on the tasks that the resource agent (henceforth bidder) would perform, i.e., bid price depends on the other bids. In that context we used the formulation proposed in [8][1] where each bid has a set of bid price modifiers which define the inter-bid dependence and allow the bidder to compact the possible combinations reducing the memory usage. Summing up, each bidder can send a set of time-dependent bids ($B_{j1} \dots B_{jN_j}$, where N_j is the number of bids sent by bidder j) together with three vectors: M_e, M_r, M_t:

$$B_{jk} = \left\langle T_i@t_{jk} : \left(CE_{jk}, CR_{jk}, CT_{jk}\right), M_{e_{jk}}, M_{r_{jk}}, M_{t_{jk}} \right\rangle \tag{4.1}$$

Where T_i is the ith task, t_{ik} is the start time proposed by the bidder, CE_{jk} the energy cost, CR_{jk} the resource cost and CT_{jk} the estimated duration of the task. Note that in the context presented in Chapter 2, the cost related to the use of a resource is not usage-dependent and so, M_r is a vector of zeros as well as M_t. Nevertheless we left them in the formalization to illustrate that they can be easily added as well as set up energy costs.

4.2 The Winner Determination Problem

We are dealing with combinatorial auctions, therefore the WDP is NP-hard [2, 8]. Nevertheless, note that we are limiting the *independency* between bids as we are using 2D matrices to compute the price of the possible combinations and so, the set up costs for combinations of three or more bids are the aggregation of the set up costs of all possible pair of bids. For example if a bidder sends three bids (we consider only one set up cost vector for simplicity):

$$B_{j1} = \left\langle T_1@t_{j1} : (20, 12, 4), [., -5, 3] \right\rangle \tag{4.2}$$

$$B_{j2} = \left\langle T_2@t_{j2} : (15, 15, 1), [-5, ., -2] \right\rangle \tag{4.3}$$

[1]They propose the formulation in the context of set up costs due to the chain of two or more tasks, but we extended its use to set the saves/extra costs over the bid prices due to the energy rates.

$$B_{j3} = \langle T_3 @ t_{j3} : (20, 13, 2), [3, -2, .] \rangle \tag{4.4}$$

These three bids are actually seven combinations of bids: B_{j1}, B_{j2}, B_{j3}, $B_{j1} + B_{j2}$, $B_{j1} + B_{j3}$, $B_{j2} + B_{j3}$ and $B_{j1} + B_{j2} + B_{j3}$. The set up costs of the last combination is $-5 - 2 + 3 = -4$.

Given the bids from all the bidders, the autioneer has to decide which is the most suitable resource agent for each task. The winners depend on the factor we want to optimize, the energy cost, the resource cost or the make span. Thus, when we want to minimize the energy costs, the winner determiniation problem is given by equation (4.5).

$$arg \min \left\{ \sum_{i=1}^{\|\mathbf{T}\|} \sum_{j=1}^{N_b} \sum_{k=1}^{N_j} \left(CE_{jk} \cdot x_{jk}^i + \sum_{l=l, l \neq j}^{\|\mathbf{T}\|} M_{e_{jk}}(l) \cdot x_{jk}^i \cdot x_{jl}^p \right) \right\} \tag{4.5}$$

where p is the task B_{jl} refers to, N_b is the number of bidders and x_{jk}^i is a binary variable that indicates whether B_{jk} is the winner bid for task i. Similarly, when we are optimizing the resource costs and the makespan, the WDP is given by:

$$arg \min \left\{ \sum_{i=1}^{\|\mathbf{T}\|} \sum_{j=1}^{N_b} \sum_{k=1}^{N_j} \left(CR_{jk} \cdot x_{jk}^i + \sum_{l=1, l \neq j}^{\|\mathbf{T}\|} M_{r_{jk}}(l) \cdot x_{jk}^i \cdot x_{jl}^p \right) \right\} \tag{4.6}$$

$$arg \min \left\{ \sum_{i=1}^{\|\mathbf{T}\|} \sum_{j=1}^{N_b} \sum_{k=1}^{N_j} \left(CT_{jk} \cdot x_{jk}^i + \sum_{l=1, l \neq j}^{\|\mathbf{T}\|} M_{t_{jk}}(l) \cdot x_{jk}^i \cdot x_{jl}^p \right) \right\} \tag{4.7}$$

Finally, the winner bids, obviously have to respect the constraints given by the workflow (start and end times, precendences, etc.) and equation (4.8) that ensures that each task has not more than one winner bid.

$$\forall_i \; 1 \leq i \leq \|\mathbf{T}\| : \qquad \sum_{j=1}^{N_b} \sum_{k=1}^{N_j} x_{jk}^i \leq 1 \tag{4.8}$$

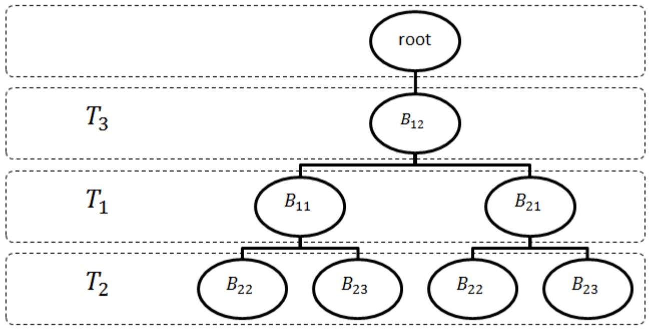

Figure 4.1: Bidtree example. Each level corresponds to a task and they are ordered into decreasing order of their demand (received bids).

As aforementioned, the WDP is a NP-hard problem and we have developed and analyzed three different methods to solve it: a Branch&Bound (B&B) algorithm and two heuristics, GA and SA.

4.2.1 Branch & Bound

Given all the bids received by the auctioneer from the bidders, we perform the search building a tree of bids (henceforth bidtree) and then searching through it the best combination of bids with a B&B algorithm.

The bidtree is a tree-shaped graph where each node corresponds to a single bid. The bidtree is divided by depth levels where each level corresponds to a single task. Therefore nodes in the same level are not connected and the tree can not be explored in a horizontal way. Figure 4.1 shows an example of bidtree where there are three auctioned tasks and two bidders that send bids $B_{11} \dots B_{12}$ and $B_{21} \dots B_{23}$. Task T_1 has two bids, B_{11} and B_{21}, T_2 has also two bids, B_{22} and B_{23}, and T_3 has a single bid, B_{12}. Since T_3 is the less bidded task, its bid is the top bid and since T_2 is the most bid task, its bids are the bottom bids. This ordering is highly important since it guarantees that the resulting tree has the minimum number of nodes.

Figure 4.2 shows the relation $\frac{N_{dec}}{N_{inc}}$, where N_{dec} is the number of nodes explored when the decreasing task order is used and N_{inc} is the number of increasing order reduces the number of explored bids. However, there is not

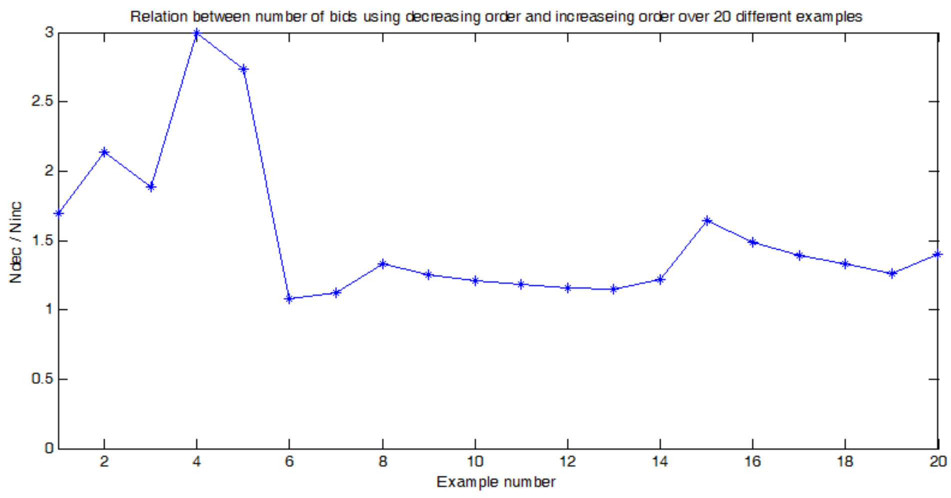

Figure 4.2: Relation between the number of nodes explored using increasing order of the tasks (as Figure 4.1) and decreasing order, $\frac{N_{dec}}{N_inc}$, over 20 different examples

an exact value for this relation since it depends on the number of bids that each task has.

Some works, [12, 4], propose also to implement a bid ordering mechanism (ordering the bids from the same level in a way that the lowest cost bids are explored first, allowing the algorithm to set better bounds) to ensure that the best combinations would be found in the early iterations. Note that this mechanism is not suitable in our case, since we have set up costs that depend on whole combination of bids and, therefore, we cannot prune a branch before exploring it till the deepest node (unless the set up cost of a pair of bids is infinity). Thus, algorithm 1 explores the tree as a depth-first-search algorithm, what allows it to keep in memory only the best branch found and the current one, pruning only those branches that have incompatible bids according to the bidtree or due to they use the same resource at the same time. Note that is not possible to make a fast estimation of the branch due to the set up costs that would be applied depend on the whole branch.

When the algorithm reaches the deepest node of the branch, it backtracks to the previous node and expands it, instead of backtracking to the root node and expanding again the branch.

Algorithm 1 BB_expand

```
Require: bag of bids (bag) grouped by tasks. Tasks ordered by decreasing order of demand.
 1: $b \leftarrow choose\,(bag\,[0])$ "take a bid of the corresponding task-level"
 2: $branch.add\,(b)$
 3: $value \leftarrow evaluate\,(branch)$
 4: $Aux \leftarrow remove\_current\_task\_bids(bag)$ "removes all bids related to the task done by b.
    Next task-level is the first."
 5: if $value < \infty \wedge \neg$leaf node then
 6:     BB_expand(Aux)
 7: else
 8:     if $value < bestValue$ then
 9:         $bestBranch \leftarrow branch$
10:     end if
11: end if
12: $branch.remove\,(b)$ "remove b from the current explored branch"
```

Note that we are not following the bidtree structure proposed in [2] because the auctioneer does not receive combinations of bids, it receive individual bids and set up costs that he has to consider when a combination of bids is proposed.

4.2.2 Genetic Algorithms

When there are a lot of tasks, bidders, etc. in the auction the WDP becomes unfeasible since it is NP-hard. Thus, we decided to analyze the performance of some metaheuristic methods on solving the WDP.

GAs, [14, 10, 6], exploit the ability of the evolution operators to improve the equality of a population of solutions generation after generation in order to find the optimum solution to a given problem.

To solve the given WDP with GA we defined the chromosomes as strings of length $\|\mathbf{T}\|$, the number of tasks of the workflow. Each slot corresponds to a task and it is assigned to a bid received by the auctioneer. The algorithm is initialized (step 1 Algorithm 2) by an amount of new chromosomes (random combinations of bids) and then they are combined (step 3 Algorithm 2) using a two point crossover, see Figure 4.3, which greatly expands the possible offspring created [6]. The parents are selected using the tournament selection using 3 candidates that tends to keep more diversity than the roulette wheel selection giving more chances to the worst chromosomes [6]. This mechanism

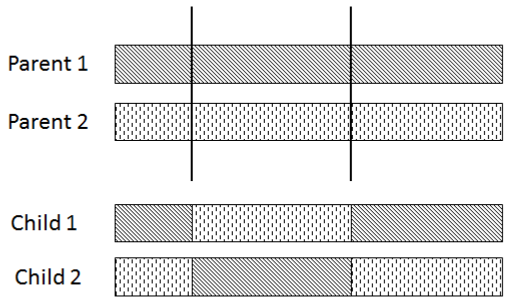

Figure 4.3: Two point crossover

consists in selecting randomly 3 chromosomes and then choose the best of them as the first parent, then three more chromosomes are selected randomly and the best is chosen as the second parent. After the children chromosomes are created, we apply a mutation operator (step 4 Algorithm 2) over them in order to increase the population diversity. The mutation operator consists in changing the chromosome's genes (changing the bit assigned to them for another one randomly selected) for a given probability $\mu = 0.1$. Once we generated the desire number of new chromosomes we lump together them with the old chromosomes and apply elitism (step 6 Algorithm 2) in order to select only the best chromosomes for the next iteration.

Algorithm 2 Genetic Algorithm

```
Require: bag of bids ($bag$) grouped by tasks, number of generations $N_g = 200$, $generation = 0$,
    population size $popSize = 100$, children chromosomes made at each iteration $N_{children}$
1: $population \leftarrow initialize\_population\,(bag, popSize)$
2: while $generation < N_g$ do
3:     Mating: Create $N_{children}$ new chromosomes using tournament selection and two point
       crossover
4:     Mutation:Apply mutation operator on each new chromosome
5:     Add new chromosomes to $population$
6:     Elitism: remove the worst chromosomes from $population$ keeping only the best $popSize$
7: end while
```

Figure 4.4 shows the evolution of the fitness of the fittest chromosome and the average of the population. Note how elitism always keeps the best chromosome and how elitism combined with crossover and mutation operators progressively improves the fitness of the population generating better chromosomes at each generation.

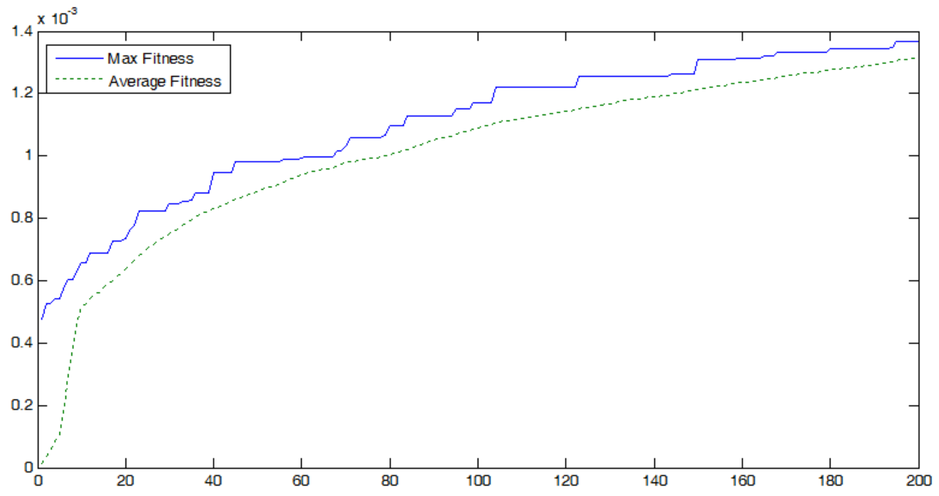

Figure 4.4: Plot of the evolution of the fitness of the best generated chromosome vs. the average fitness of the whole population. The fitness is computed as $\frac{1}{cost}$

4.2.3 Simulated Annealing

SA, [13, 9], is a metaheuristic method inspired on a metallurgy technique that heats and cools the material to move their atoms in order to allow them achieving lower energy states. SA (see Algorithm 3) tries to iteratively improve an initial solution with this heating-and-cooling process by selecting at each iteration a neighbor solution. If the new solution is better that the old one, then the algorithm moves towards the new solution, otherwise it moves towards the new solution with a probability of $e^{\frac{v_{old}-v_{new}}{T}}$ or stays at the old solution with a probability $1-e^{\frac{C_{old}-C_{new}}{T}}$, where C_{old} and C_{new} are the costs of the old and new solution respectively, and T is the temperature of the algorithm. Thus, when SA finds a worse solution it has a chance to move towards it (to make a bad move) in order to avoid getting stacked on local optimums or flat regions. The temperature of the algorithm T is progressively reduced, reducing so the chances of bad movements at each iteration. Figure 4.5 shows the solutions explored by SA, the solutions where it decides to move or stay and the best solution found since the beginning

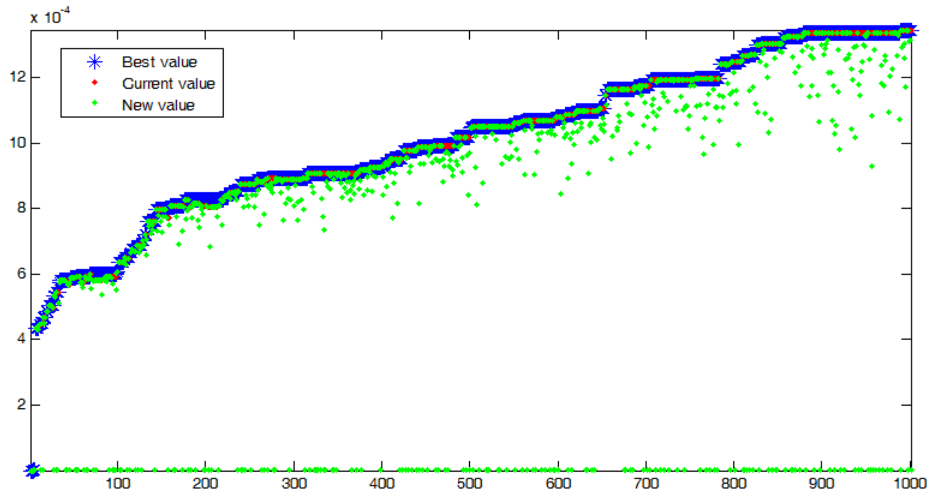

Figure 4.5: Plot of the solutions explored by SA. We can see the best solution found since the start of the algorithm, the current solution where the algorithm stays and the new solutions explored by the algorithm. We plotted $\frac{1}{cost}$ to set to zero those solution with an infinity cost.)

at each iteration which is always saved and replaced when a better one is found.

Solutions are equal to the chromosomes explained in the previous subsection. They are strings where each slot corresponds to a task of the auctioned workflow and each slot has assigned a single bid. When the algorithm selects a neighbor solution from another solution, it assigns a probability of being changed to each bid of the solution. This probability is proportional to an estimation of its contribution to the cost of the solution. Thus, cheaper bids has less chances of being changed.

Algorithm 3 Simulated Annealing

```
Require: bag of bids ($bag$) grouped by tasks, $T_o = 1000$, $\delta = 0.99$, $T_f = 3.5 \times 10^{-15}$
 1: set an initial solution $s$ randomly
 2: $C_{old} \leftarrow evaluate\,(s)$
 3: $s_{best} \leftarrow s$
 4: $C_{best} \leftarrow C_{old}$
 5: $T \leftarrow T_o$
 6: while $T > T_f$ do
 7:     $s_{new} \leftarrow neighbor\_solution\,(s, bag)$
 8:     $C_{new} \leftarrow evaluate\,(s_{new})$
 9:     $r \leftarrow$ random number between 0 and 1
10:     if $C_{new} < C_{old} \vee r < e^{\frac{C_{old} - C_{new}}{T}}$ then
11:         $C_{old} \leftarrow C_{new}$
12:         $s \leftarrow s_{new}$
13:         if $C_{old} < C_{best}$ then
14:             $C_{best} \leftarrow C_{old}$
15:             $s_{best} \leftarrow s$
16:         end if
17:     end if
18:     $T \leftarrow T \times \delta$
19: end while
20: return $s_{best}$
```

Chapter 5

Results & Discussion

In previous chapters we defined the scheduling problem and we presented different methods for solving it. Here we present the results obtained by the different presented methods.

Figure 5.1 shows us the elapsed time by Choco solver to solve the scheduling problem and the elapsed time by bidtree B&B, GA and SA to solve the WDP. We can state that since the constraint programming problem and the WDP are NP-hard, Choco and bidtree B&B solve them into an exponential time. However, B&B solves the problem faster than Choco what makes us think that solving the scheduling problem by auctions plus bidtree B&B is more efficient than using Choco. Moreover, the auctions mechanism offers other advantages. For example we can model the participation of several independent agents or organizations which do not want to share all the information.

On the other hand, Figure 5.1 shows that the elapsed time by SA and GA does not grows up exponentially with the number of auctioned tasks. This makes them, a priori, suitable methods for solving big problems. However, if we focus on Figure 5.2 we notice that GA and SA are not able to find, always, the optimum solution (as we expected) since they are not complete optimization algorithms. Also the chances SA or GA have to achieve the optimum solution decrease according the number of auctioned tasks increases. Therefore, there is a trade-off between time and quality. This trade-off is

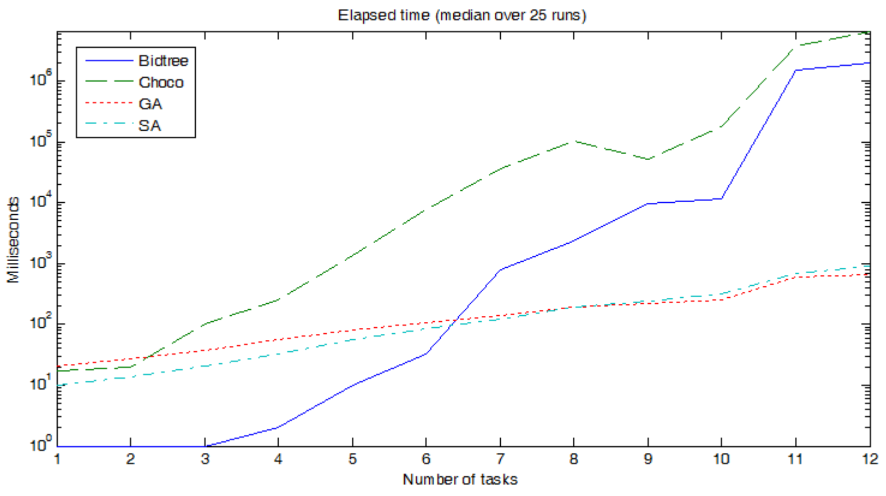

Figure 5.1: Elapsed time for solving the scheduling problem by Choco solver or the WDP by bidtree B&B or GA or SA (median over 25 runs)

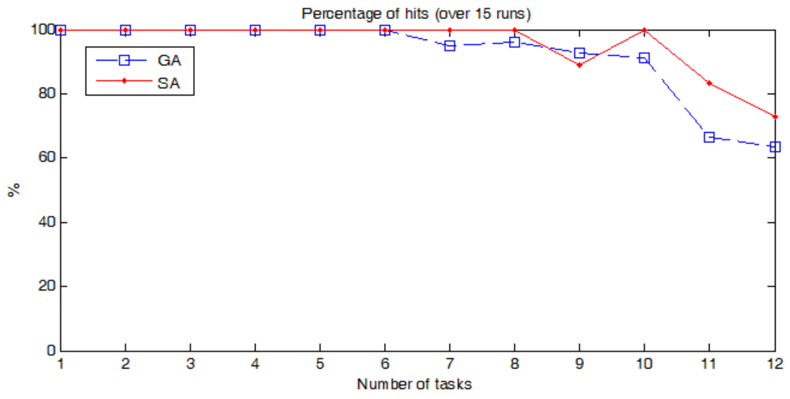

Figure 5.2: Percentage of times SA and GA find the optimum solution

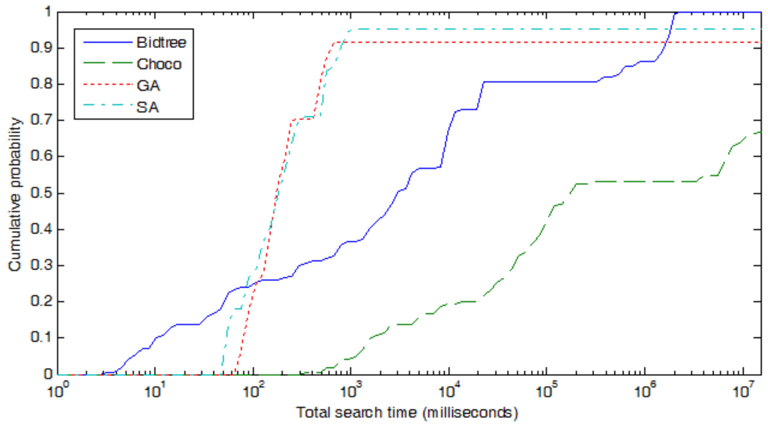

Figure 5.3: Performance of bidtree B&B, Choco, GA and SA along 150 random problems with $4 - 12$ tasks each one.

also represented on Figure 5.3 that shows the probability of achieving the optimal solution given an amount of time. Furthermore it illustrates that GA and SA never have 100% of chances to achieve the best solution, but bidtree does if enough time is given.

To compare the performance of GA and SA we extended the experimentation with longer workflows (from 4 to 23 tasks each workflow). Glancing at Figure 5.4, which shows the cumulative probability of getting a feasible solution given an amount of time, we can state that GA has a faster convergence than SA, but both have similar probability of achieving a feasible solution. Moreover, when both algorithms find a feasible solution, 20.09% of times GA's solution is better than SA's, and 12.36% of times SA's solution is better than GA's. The other times, both achieves the same solution. Therefore, focusing on the quality of the given solution, both algorithms perform similarly, with a light advantage of GA in scenareos with longer workflows. But in time terms, GA has a faster convergence which means it is faster.

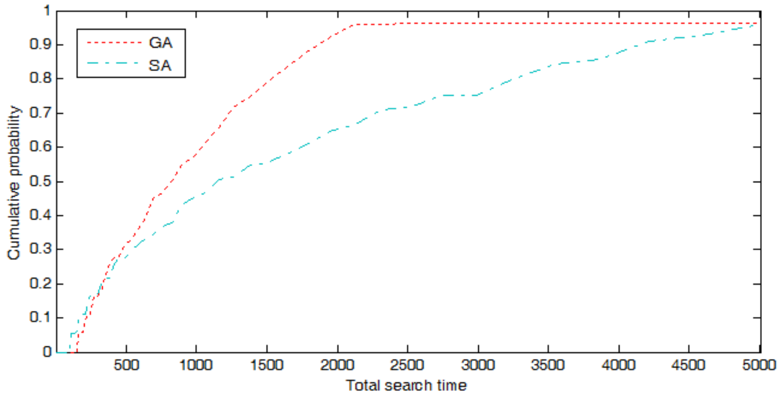

Figure 5.4: Performance of GA and SA along 250 random problems with $4 - 23$ tasks each one.

Chapter 6

Conlusions & Future work

In this work we stated the scheduling problem when time-dependent costs are present and we proposed solving it by a constraint programming model and an auction mechanism. We solved the constraint programming model by Choco solver and the winner determination problem using a bidtree branch&bound algorithm, genetic algorithms and simulated annealing. We have seen that the auction mechanism present some advantages like it is able to model the problem when there are several independent organizations involved in the problem solving. Moreover, the results we have presented demonstrate that the use of auctions with the bidtree branch&bound algorithm is more efficient than solving the constraint programming model with Choco. However, the elapsed time by Choco and the branch&bound algorithm for solving both problems grows up exponentially with the number of tasks the problem has. Then its use is tied up to the dimensionality of the problem. Furthermore, we apply two heuristic methods for solving the winner determination problem, genetic algorithms and simulated annealing. The results showed us that this algorithms can be used to solve big problems, however they do not find always the optimal solution, and their chances for doing so decrease as the complexity of the problem grows up.

Given the results presented in this work, it would be interesting to analyze how the lack of optimality of the solutions found by the heuristic algorithms affects the resource agents behavior. We also presented different different target functions depending on what is desired to optimize (energy costs, resource costs, makespan), thus it would be interesting to complement this work studying multi attribute auctions.

Acknowledgments.

This research project has been partially funded through BR-UdG Scholarship of the University of Girona granted to Ferran Torrent-Fontbona.

Give special thanks to Albert Pla for sharing his auction simulator.

Bibliography

[1] Indranil Bose and Raktim Pal. Do green supply chain management initiatives impact stock prices of firms? *Decision Support Systems*, 52(3):624–634, 2 2012.

[2] John Collins, Güleser Demir, and Maria Gini. Bidtree ordering in ida combinatorial auction winner-determination with side constraints. *Agent-Mediated Electronic Commerce IV. Designing Mechanisms and Systems*, 2531:17–33, 2002.

[3] Red Eléctrica de España. BoletÃŋn mensual diciembre 2012, 14th February 2013 2013.

[4] Laureano F. Escudero, Mercedes Landete, and Alfredo MarÃŋn. A branch-and-cut algorithm for the winner determination problem. *Decision Support Systems*, 46(3):649–659, 2009.

[5] Sebastian Gottwalt, Wolfgang Ketter, Carsten Block, John Collins, and Christof Weinhardt. Demand side management - a simulation of household behavior under variable prices. *Energy Policy*, 39(12):8163–8174, 12 2011.

[6] Randy L. Haupt and Sue Ellen Haupt. *Practical genetic algorithms.* Wiley-Interscience, 2004.

[7] Simonis Helmut and Tarik Hadzic. *A Resource Cost Aware Cumulative*, pages 76–76–89. ecent Advances in Constraints. Springer Berlin, Heidelberg, lecture notes in computer science edition, 2011.

[8] Beatriz Lopez, Aditya Ghose, Bastin Tony Roy Savarimuthu, Mariusz Nowostawski, Michael Winikoff, and Stephen Cranefield. Energy-aware optimisation of business processes. 2012.

[9] Sean Luke. Essentials of metaheuristics. *Lecture notes, George Mason University.Free access: http://cs.gmu.edu/ sean/book/metaheuristics/(last visited: 2011.11.16)*, 2009.

[10] Mitchell Melanie. An introduction to genetic algorithms. *Cambridge, Massachusetts London, England, Fifth printing*, 1999.

[11] Albert Pla, Beatriz Lopez, and Javier Murillo. Multi-attribute auction mechanism for supporting resource allocation in business process enactment. *STAIRS 2012*, pages 228–228–239, 2012.

[12] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Cabob: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51(3):374–390, 2005.

[13] F. Torrent-Fontbona, V. Muñoz, and B. Lopez. Solving large immobile location-allocation by affinity propagation and simulated annealing. application to select which sporting event to watch. *Expert Systems with Applications*, (0), 2013.

[14] Ferran Torrent-Fontbona. Decision support methods for global optimization. Master's thesis, University of Girona, 2012.

[15] M. Wieczorek, S. Podlipnig, R. Prodan, and T. Fahringer. Applying double auctions for scheduling of workflows on the grid. In *2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2008*. Institute of Computer Science, Austria, 15th-21st November 2008.