PROCEEDINGS OF SCIENCE

Data Transfer Optimization: Going Beyond Heuristics

Roman Barták[1]
Charles University in Prague, Faculty of Mathematics and Physics
Malostranské náměstí 2/25, 118 00 Praha 1, Czech Republic
E-mail: bartak@ktiml.mff.cuni.cz

Scheduling data transfers is frequently realized using heuristic approaches. This is justifiable for on-line systems when extremely fast response is required, however, when sending large amount of data such as transferring large files or streaming video, it is worthwhile to do real optimization. This paper describes formal models for various networking problems with the focus on data networks. In particular we describe how to model the path placement problems where the task is to select a path for each demand starting at some source node and finishing at the destination node. The demands can be instantaneous, for example in data streaming, or spread in time in the so called bandwidth on demand problems. This second problem is a complex combination of path placement and cumulative scheduling problems. As constraint programming is a successful technology for solving the scheduling problems, we sketch the basic principles of constraint programming and illustrate how constraint programming can be applied to solve the above mentioned problems. The main advantage of this solving approach is extendibility where the base model can be augmented with additional constraints derived from the specific problem requirements.

13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research - ACAT 2010
Jaipur, India
February 22–27 2010

[1] Speaker
The author is supported by the Czech Science Foundation under the contract P202/10/1188.

PoS(ACAT2010)009

1. Introduction

Optimization is becoming a keyword in many aspects of current business and industry. The IT companies started to pay attention to energy consumption of data and supercomputer centres and attempted to minimize it with the goal to decrease their expenses. The airlines have been for a long time maximizing utilization of their most valuable assets, that is, airplanes and crew. The manufacturing companies are trying to minimize waste during production and increase throughput of the factories to speed up production while minimizing expenses. Optimization aspects can be found in almost any area of human endeavour even if the background reasons could be as different as increased competition, global economic crisis, or green policy.

Many of the above optimization efforts use ad-hoc techniques based on long experience with a given application area. This surely brings some immediate effects but this approach also has some significant drawbacks in the long term. The main disadvantage is bad flexibility of ad-hoc techniques that is especially critical in current complex dynamic environments. The nature of ad-hoc techniques is that they work for a given problem. If the problem changes, sometimes even slightly, then the ad-hoc optimization technique may not work anymore and a completely new technique must be found. A typical example could be including a new resource, for example a fast computer, to an existing system that requires redesign of the ad-hoc task allocation algorithms to assume this resource. This *dynamic inflexibility* means that it is necessary to develop new optimization techniques (or to manually tune the existing techniques) as new problems are coming. The inflexibility of ad-hoc techniques also makes their usage hard in structurally complex environments. It is possible to optimize one part of the system, for example network throughput, but if this optimization is not in accordance with the optimization of other parts of the system, for example utilization of CPUs, then the overall effect may be even the decrease of performance. We can call it bottleneck shifting as optimization of one part of the system could make the bottleneck in another part of the system that was working fine so far. This *structural inflexibility* makes it hard to optimize complex systems even if good ad-hoc techniques for optimizing particular components may be known and widely used.

Due to the above reasons, we argue for using "real" optimization techniques that are based on a formal model of the problem and generally applicable global optimization frameworks. It does not mean that the current expertise with solving the problem is completely neglected, but it appears in a less direct form, for example, as a heuristic guiding the optimisation procedure to locally select between the alternative solutions. We can say that the formal model and the general optimization framework provide the global view of the problem while the existing ad-hoc techniques may serve as local guides when solving the problem. Such optimization approach is probably harder to implement at the initial stage but it pays off in the long term due to increased flexibility and exploitation of development in the large area of optimization techniques.

In this paper we will provide a formal view of one area of high performance computing that is data transfer optimization. We will present formal models for the path-placement

problem which is about finding a path to transfer data from source to destination. This problem will serve us to demonstrate that there could be different formal ways to see a single problem. We will also show how to modify the model to solve a closely related bandwidth placement problem to demonstrate the flexibility of the models. This part is based mainly on the survey paper by Helmut Simonis [1] with some comments from our own experience in implementing the models. In the second part, we will explain the details of one of the possible optimization technologies applicable to data transfer optimization, namely Constraint Programming. This technology is based on integrating inference and search and it is in particular appropriate for problems where it is hard to find a feasible solution and where there are many constraints.

2. Formal Models for Networking Problems

There are various types of networking problems and networks with specific features. For example, planning water supply must assume leaking pipes (many existing pipes are quite old) and different quality of water from different sources while planning electricity distribution must assume that the electrical networks have very limited capabilities to store electricity in the distribution nodes. In this section we shall focus on data networks, that is, networks used to transfer data between the storage and computing nodes. There are various problems related to data networks starting with the network design problem dealing with defining connectivity and finding capacity of links to satisfy an expected set of demands. In this paper we assume that the network infrastructure is given and we focus on planning the paths through the network to move data from source to destination. This path placement problem is defined by a set of demands – a set of requests to move data from their current location to another location – and the task is to find a path for each demand respecting the capacity limits of network links. We will present the model for the pure placement problem and then we will show how easy it is to modify the model for a problem where the time of demands is assumed – a so called bandwidth placement problem.

2.1 Modelling Approaches

The problems studied in this paper are about the paths in the graphs and hence the critical modelling decision is how to formally describe a path. There are three main approaches usable to formally model the paths.

- The **path-based model** assumes that for each demand we know all possible paths in the network. Hence, for each demand and for each path we can use a decision variable describing whether the path is used for that demand or not.
- The **link-based model** is more fine-grained as it uses a decision variable for each demand and for each link in the network to describe whether the link is used for the demand. The links must be connected to form a path which is done via additional constraints.
- Finally, the **node-based model** describes how (and if) each demand leaves each node. In other words, there is a decision variable for each demand and for each node describing the successor node in the path going through the node (the nodes can be identified by natural numbers where value 0 means “no successor node”). Again, some additional constraints are necessary to ensure that a path is formed in the graph.

There are some principal differences between the above models. The path-based model requires all possible paths to be known in advance which could be a problem if there are many alternative paths (more paths imply more decision variables). Hence, this model can hardly be directly implemented. Moreover, this model is designed to find paths and it is more complicated to model a more complex structures of data transfers such as trees (more sources and a single destination and vice versa). The link-based model requires some additional constraints in comparison to the path-based model to ensure that the used links form a path. These constraints are typically expressed in the form of Kirchhoff's Current Law. Briefly speaking, if the path enters some intermediate node, it must leave it as well. We will show a formal model later. On the other hand, the link-based model provides more flexibility for describing other data-transfer structures such as trees. This can be achieved by simple modification of the constraints describing the Kirchhoff's Current Law [2]. Finally, the node-based model assumes a single outgoing link for each node and demand (as we described the model) so again it imposes some restrictions on the supported types of data-transfer structures (but the tree can be easily modelled). Similarly to the link-based approach, it also requires additional constraints to ensure that the data transfer is not interrupted (if data are entering the intermediate node, they must also leave it).

From the above analysis, the node-based approach seems to be the most flexible one and we will give more details about this model in the next sections. Nevertheless, let us first briefly show where and how the other modelling approaches were used. The path-based model has been used in [3] where the column generation technique has been applied. This mathematical programming technique is a natural solving approach for this model as only a subset of paths is considered. The objective function can drive the search for finding new paths improving the overall solution quality. The constraint programming oriented view of the path-based approach has been studied in [4] where a method of blocking islands was used as an inference technique. The blocking island of size *d* is a set of nodes such that all nodes in the set can reach all other nodes in the set using the paths of capacity at least *d*. If some path is placed then the blocking islands are updated to find out which demands can still be satisfied. We did some comparison of path-based and link-based models in [5] where constraint programming was used as the underlying optimisation technology. The conclusion from the experiment was that the link-based model produced significantly better results than the path-based approach.

The node-based approach was proposed in [6] for solving the Bandwidth on Demand Problem (see below). They modified the model in such a way that the last node in the path is connected to the first node to form a cycle. This is a typical way how to represent the path in the node-based approach as it allows using a dedicated global constraint *cycle* [7] to cover the network by a set of cycles (see below for information about global constraints). Still such a model requires some dummy nodes and links to connect the other (non-used) nodes into another cycle. As argued in [1], this is not the best choice as it hinders inference. In [8] we used the node-based and link-based models for a problem of finding a path for a robot which can be seen as a variant of the path-placement problem. Again, we applied the constraint programming approach, but instead of the *cycle* constraint we used a more general and widely available *global cardinality constraint* (see below) [9] and a single dummy node in the node-based model. Briefly speaking, the global cardinality constraint can restrict how many links point to given

nodes (to form cycles, at most one link can enter any node). The preliminary experiments showed that if the problem is highly constrained then the node-based approach is in fact more efficient than the link-based approach.

In the remaining sections we will focus on the node-based models and how they describe particular networking problems.

2.2 Path Placement Problems

The Path Placement Problem can be characterised as deciding which path is assigned to each demand under the capacity constraints of the network. We will use the following notation in the formal model. Let D be the set of demands, N be the set of nodes and E be the set of links (directed edges) in the network. We will denote IN(n) the set of links entering the node n and OUT(n) the set of links leaving the node n. Each edge e∈E has its capacity cap(e). For each demand d∈D, let dest(d) ∈ N be the destination node of the demand and orig(d) ∈ N be the source node of the demand. Each demand d ∈ D has some non-negative value val(d) and it requires some bandwidth bw(d) of each link that it uses. The capacity constraint says that the sum of bandwidths of all demands using a given link does not exceed its capacity

When assigning the path, it is possible to optimize various utility functions. Probably the most straightforward version of the problem is a so called *demand acceptance problem* where the task is to select a set of demands with the maximal total value and assigning a path to each selected demand such that the paths satisfy the capacity constraints of the network. It means that for each demand d we can use a 0-1 decision variable Z_d which indicates whether the demand is accepted or not. In the link-based model there are also decision variables X_{de} indicating whether the demand d is routed over the edge e in the network. The following MILP (Mixed Integer Linear Programming) model describes the problem formally.

$$\max_{\{Z_d, X_{de}\}} \quad \sum_{d \in \mathbf{D}} \mathtt{val}(d) Z_d$$

under the constraints

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N}: \quad \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \mathtt{dest}(d) \\ Z_d & n = \mathtt{orig}(d) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall e \in \mathbf{E}: \quad \sum_{d \in \mathbf{D}} \mathtt{bw}(d) X_{de} \leq \mathtt{cap}(e)$$

$$Z_d \in \{0,1\}$$

$$X_{de} \in \{0,1\}$$

There are two main constraints used in the model. The *path constraint* (first) ensures that that there is a single path for each accepted demand. As already mentioned the constraint expresses the Kirchhoff's Current Law – the input to the intermediate node equals the output; there is only output from the source node (if the demand is accepted) and only input to the destination node (if the demand is accepted). The *capacity constraint* (second) guarantees that the total capacity of demands routed over each edge does not exceed its capacity.

Frequently, it is required that all demands are accepted and link or network utilization is being optimized. This is called the *traffic placement problem*. Clearly, variables Z_d are no more required (they are all equal to 1) and the only decision variables remain the variables X_{de}. However there seems to be no consensus about the objective function for this type of problems. For example the minimization of maximal utilisation of links can be expressed as:

$$\min_{\{X_{de}\}} \max_{e \in \mathbf{E}} \frac{1}{\mathtt{cap}(e)} \sum_{d \in \mathbf{D}} \mathtt{bw}(d) X_{de}$$

The minimization of overall network utilisation is expressed as:

$$\min_{\{X_{de}\}} \sum_{e \in \mathbf{E}} \sum_{d \in \mathbf{D}} \mathtt{bw}(d) X_{de}$$

Another alternative was proposed in [10] where the average link utilisation is being minimized:

$$\min_{\{X_{de}\}} \sum_{e \in \mathbf{E}} \sum_{d \in \mathbf{D}} \frac{\mathtt{bw}(d)}{\mathtt{cap}(e)} X_{de}$$

2.3 Bandwidth Placement Problems

One of the advantages of the optimization techniques is flexibility where the model can be easily modified when the problem changes without changing the optimisation algorithms. Note that it is usually easier to change the declarative model than to change the solving algorithm. We already presented several model versions in the previous section. All of them assumed that all demands were simultaneous. In this section we will show, how to modify the model when each demand d has a fixed start time start(d) and fixed end time end(d). It means that the demand consumes the capacity of links only at the specified time interval. In other words, two demands interact only if they overlap in time. This is called a *Bandwidth Placement Problem* and it requires that at any time point the network capacity is not exceeded. Fortunately, it is not necessary to check this constraint at any time point but only at the time points when the demand on the network increases. These are exactly the start times of the demands. Let T = {start(d) | d∈D} be the set of such time points. Then the MILP formal model looks as follows:

$$\max_{\{Z_d, X_{de}\}} \quad \sum_{d \in \mathbf{D}} \mathtt{val}(d) Z_d$$

under the constraints

$$\forall d \in \mathbf{D}, \forall n \in \mathbf{N}: \quad \sum_{e \in \mathbf{OUT}(n)} X_{de} - \sum_{e \in \mathbf{IN}(n)} X_{de} = \begin{cases} -Z_d & n = \mathtt{dest}(d) \\ Z_d & n = \mathtt{orig}(d) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall t \in \mathbf{T}, \forall e \in \mathbf{E}: \quad \sum_{\substack{d \in \mathbf{D} \\ \mathtt{start}(d) \le t \\ t < \mathtt{end}(d)}} \mathtt{bw}(d) X_{de} \le \mathtt{cap}(e)$$

$$Z_d \in \{0,1\}$$

$$X_{de} \in \{0,1\}$$

Notice that only the capacity constraint has been modified in the model. This constraint is known in constraint programming as the cumulative constraint [11] and it can be used even if the position of demand in time is not fixed a priori but it is decided when solving the problem. Such problem then becomes a difficult combination of path placement and cumulative scheduling problems.

An on-line extension of the bandwidth placement problem was studied in [12] where new demands are added as time passes while the previous commitments, that is, accepted demands, must be respected. The authors used a constraint-based conflict resolution mechanism when a new demand cannot be added to the network. In such a situation the constraint model is built to find a new path for accepted demands, which can be seen as repairing the previous solution. Note that the already accepted demands must remain accepted but their routing may change.

All above mentioned problems assumed that all links associated with the demand are used all the time when the demand is active. This is true for example for videoconferencing but when files are moved through the network it is possible that only the link where the file is currently being routed is occupied. Moreover, the file can be stored in the nodes for some time before it continues to the next link. Finally, the typical objective in such a problem is having the files as soon as possible in the destination – a so called makespan. In [5] we suggested to split the above model to the planning stage, where the links are associated with the demands, and the scheduling stage, where the time of using the link is decided. The reason for this decomposition is that the second problem is a classical scheduling problem and can be solved fast using the state of the art constraint-based techniques. Opposite to the above models, we did another important assumption. Rather than allowing several demands to be routed through a single link at the same time while respecting the capacity constraints we assume that the demand fully occupies the link and hence the link is seen as a unary (or disjunctive) resource [11]. In such a case we are talking about the transfer activity whose duration can be computed from the speed of the link and the size of transferred data. Figure 1 gives an example how the transfer of two files sharing the same link looks when modelled as a scheduling problem.

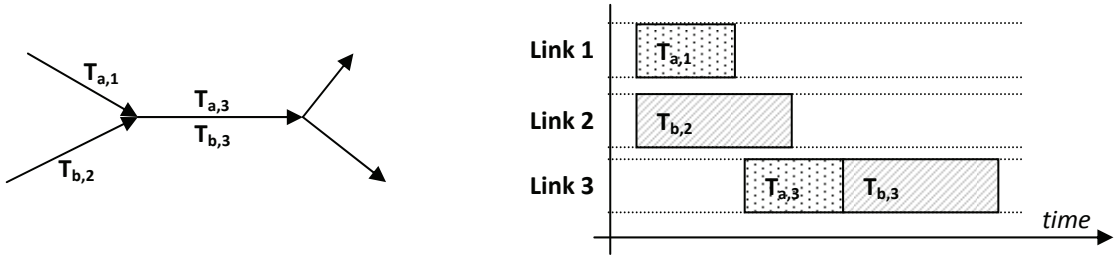

Figure 1: Transfer of files (a and b) through a shared link (left) can be seen as a scheduling problem of allocating transfer activities to unary resources that represent the links (right).

Using unary resources eliminates the necessity to do complex cumulative scheduling as there are many efficient inference algorithms behind the unary resource constraints [11,13,14]. Naturally, the model with unary resources is just an approximation of reality, but our results in [5] showed that the difference of makespan computed by our model from the real makespan (found by the simulation of real networks) is less than 3% which is negligible.

3. Technology Background

In the previous sections we described several models formalizing particular networking problems, but to find the solution, one needs the appropriate optimisation technology behind the models. There exist several approaches for solving combinatorial optimization problems including the data placement problems. On one side there are ad-hoc techniques that include both exact algorithms and heuristic approaches. The advantage of these methods is high-customization to a particular class of problems which for example can guarantee the runtime or the quality of the solution. This high-customization is also a disadvantage of these methods because it is more demanding for the development time (a solving algorithm must be developed) and if the problem changes, sometimes even only slightly, then the used algorithm may not be applicable anymore and a new solver must be programmed. On the other side, there are general solving techniques such as mathematical programming that requires the problem to be formulated (modelled) in a particular formalism, for example, as a set of linear inequalities, and then uses a general solving algorithm for this formulation. The advantage of these techniques is using a more declarative approach to problem solving based on modelling the problem formally rather than writing a solving algorithm. It is usually easier to change the model than to change the solving algorithm if the problem changes. This approach also exploits the work done in the implementation of the general solvers so if a faster solver is available then the whole system becomes faster without any change of the model. On the other hand, the general solvers may not achieve the efficiency of ad-hoc algorithms as they are designed to solve a wider range of problems. In this paper we advocate for using this second approach where the problem is declaratively modelled and then a general solver is applied to the model.

The models presented for the networking problems were in purpose specified as systems of linear inequalities. This formulation is appropriate for mathematical programming methods such as *Mixed Integer Linear Programming* (MILP). MILP problems are solved by the mechanism that can be characterized as the combination of search and relaxation. Relaxation means that some constraints are relaxed in the problem; in this case, the integrality constraints requiring some variables to be integers are removed, to make the problem polynomially solvable. The solution of the relaxed problem can then be used to estimate the value of the objective function and to restrict the area where the optimal integer solution is located. Search techniques are then used to cut off the non-integer solutions for example by adding new constraints. This process is repeated until an integer solution is found. This approach is in particular appropriate for the optimisation problems, but it requires the constraints to be expressed as linear inequalities.

At the end of the previous section we described a model that uses constraint-based scheduling technology. *Constraint Programming* (CP) is a one of the younger representatives of general problem solvers for combinatorial optimization problems. It accepts constraints in any form, even ad-hoc constraints described as a set of value tuples satisfying the constraint can be used. The CP solving technology originates in Artificial Intelligence and it can be characterised as the combination of search and inference. Inference techniques are used to remove values violating some constraints while the search algorithms typically explore the possible instantiations of the decision variables. Thanks to strong inference techniques, CP is appropriate

for the problems with many hard constraints making the feasibility region small. Moreover CP is a flexible modelling and solving technology. It can easily accommodate various types of constraints and the user can influence the solving algorithm by the problem-dependent heuristics. We will give more details about the constraint-based solving techniques now.

3.1 Constraint Programming at a Glance

Constraint satisfaction is a technology for solving combinatorial (optimization) problems. A *constraint satisfaction problem* (CSP) is a triple (X, D, C), where X is a finite set of decision variables, for each $x_i \in X$, $D_i \in D$ is a finite set of possible values for the variable x_i (the domain), and C is a finite set of constraints [15]. A constraint is a relation over a subset of variables (its *scope*) that restricts the possible combinations of values to be assigned to the variables. A *solution to a CSP* is a complete instantiation of variables such that the values are taken from the respective domains and all the constraints are satisfied. As demonstrated in the previous sections, the decision variable may describe whether a given demand uses a given link or not. Then the domain of the variable contains two values. For constraint satisfaction it is more typical to have domains with more than two elements. The constraints restrict the combinations of values assigned to the variables in the feasible solutions. For example, if the file enters some node (one of the decision variables in the input links equals 1) then the file must leave the node (one of the decision variables in the output links also equals 1). The nice feature of CP is that the constraints can be expressed in many forms (see below).

As we already mentioned CSPs are usually solved by the combination of inference techniques and search. The major inference technique is *(generalized) arc consistency* which ensures that each constraint is locally (arc) consistent. We say that the constraint is *arc consistent* if for any value in the domain of any variable in the scope of the constraint there exist values (called a *support*) in the domains of the remaining variables in the constraint's scope such that the value tuple satisfies the constraint. For example, if the domains of variables X and Y are {1,2,3} then the constraint $X < Y$ is not arc consistent because the value 3 for X has no support in the domain of Y. To make the constraint consistent, it is enough to remove the unsupported values, which is done by a *filtering algorithm* attached to each constraint. The constraint $X < Y$ in the previous example is made arc consistent by removing 3 from the domain of X and 1 from the domain of Y.

The constraints can be expressed in many ways. The extensional representation uses a set of value tuples that satisfy the constraints. Such constraints are called *tabular* or *ad-hoc*. Tabular constraints can describe any relation, but the disadvantage could be the size of the extensional representation that needs to be explored by the filtering algorithm. Constraints can also be expressed in the form of an arithmetical or a logical formula similarly to models in this paper. The filtering algorithm behind such constraints can exploit their semantics and hence its time and space complexity are usually better than using the set of all compatible value tuples. For example the filtering algorithm for the constraint $X < Y$ does not need to check whether all values in the domains of the variables have some support or not. It is enough to ensure that the minimal value in the domain of Y is greater than the minimal value in the domain of X and

similarly for the maximum values. We expressed all constraints in the previous sections as arithmetic constraints.

There are also combinatorial constraints where the semantic is given by some underlying combinatorial structure. These constraints are frequently called *global constraints*. For example, the *all-different* constraint [16] requires all variables in its scope to have different values. A generalised version of this constraint called *global cardinality constraint* [9] allows specifying how many times a given value is assigned to the variables in its scope (the all-different requires that each value is used at most ones). This type of constraints can be used to restrict how many times a node is visited in the path placement problems as we did it in [8]. There is another global constraint that can be used to describe the path placement problems. The network can be seen as a finite state automaton and the path from source to destination corresponds to the word accepted by the automat. The set of valid paths can then be described as a regular language. We used this approach in [8]. There exists a *regular* constraint [17] that requires the values assigned to the variables in its scope (the variables are ordered) to form a word that is accepted by a given finite state automaton. Hence a single *regular* constraint can describe a valid path in the network. Actually, in [8] we used a different formulation of this constraint originated in [18] and generalised in [19] called the *slide* constraint. This meta-constraint is defined over a sequence of variables and a given k-ary constraint C in the following way: the constraint C is applied to all k-tuples of neighbouring variables in the sequence (C slides over the sequence of variables). The regular constraint can be modelled as a slide constraint where C describes the allowed transitions between the states.

In addition to general global constraints applicable to many different problems, application-dependent global constraints were also proposed. As scheduling is the show room for the CP technology many scheduling global constraints were proposed in particular to describe the resources. There exist several global constraints describing the unary resources (sometimes also called disjunctive resources) [11,13,14], that is, the resources that can process at most one activity at given time. These constraints differ in the level of inference they achieve. As showed in [5] we can approximate the network links using unary resources so these constraints are valuable for path placement problems. We already mentioned the cumulative constraint that models so called cumulative resources [11]. The cumulative resource can process several activities at the same time while its capacity cannot be exceeded. We can use the cumulative constraint for example to describe the storage resources (the storage activity starts when data are placed to the storage and it finishes when data are deleted; the consumed capacity naturally specifies the quantity of stored data). Cumulative resources can also be used to approximate a set of unary resources. For example assume that we have a set of identical CPUs, each can process a single task at given time, but it is not necessary to know to which CPU the task is allocated. This CPU bunch can be naturally modelled as one cumulative resource where the available capacity corresponds to the number of individual CPUs. Existence of these application-dependent global constraints significantly simplifies problem modelling.

We already showed that the same problem can be described using different decision variables and constraints. For example, the link-based model uses Boolean decision variables specifying whether the link is allocated to the demand or not, while the node-based model uses multi-valued decision variables describing the next node (if any) in the path. The choice of

decision variables influences how the constraints are expressed. Clearly, the level of inference (the number of deleted incompatible values) depends on the constraints in the model. Hence the choice of proper constraint model is critical for efficient solution of the problem.

So far we focused on the inference part of constraint solvers, that is, on the specification of the constraint model (what the decision variables, their domains, and constraints are). Note that there also exist inference techniques stronger than arc consistency [15], but arc consistency is the most frequently used due to a good ratio between the inference power and time and space complexity. The second component of most constraint solvers is a search algorithm that, briefly speaking, explores the possible instantiations of the variables. This can be done by selecting a variable whose domain (after applying the inference techniques) is still not singleton and selecting a value from the current domain that is then assigned to the variable. If this instantiation is proved to be infeasible (immediately by applying the inference or later by search) then another value is tried and so on until the feasible solution is found or it is proved that there is no solution (then backtracking occurs to the previously instantiated variable, if any). Such search framework can be easily customized by heuristics which recommend the variable to be instantiated first and the value to be assigned to this variable. This is a way how existing greedy techniques can be utilised within the constraint satisfaction framework. For example, in the link based model, one can select the fastest available link (variable selection) and assign it to the earliest demand (value selection). These search heuristics are a natural way how the past experience from the problem can be encoded in the general solving mechanism.

4. Summary and Vision

In the paper we presented several formal models for describing data transfer problems and we argued for applying general optimisation techniques such as Constraint Programming to solve these models. Deciding about the (sub-)optimal path for transferring data is just a first step in a more complex process of complete scheduling of scientific workflows. A typical reason to transfer data from some repository to a distant computer is using that data to do some computation with them. Frequently, the users do not care where the actual computation happens; they just need the results as soon as possible. Hence we can see the process of optimizing scientific workflows as a traditional scheduling problem with a specific type of resources. The users specify the required processing steps (actions) for their data together with the resource requirements (CPU, storage etc.) and define the particular data to be processed. The task of the planning/scheduling system is to decide where and when the particular processing steps will be realized and how to transfer data there and the results back to the user. As scheduling is an area where Constraint Programming belongs to the most successful solving technologies, we strongly believe that its power may significantly contribute to solving hard optimization problems in the area of high performance computing and make the above vision of automated data processing a reality. We presented the preliminary results in [2,5] showing that these optimisation techniques can indeed improve quality of obtained solutions.

References

[1] H. Simonis, Constraint Applications in Networks, in *Handbook of Constraint Programming*, Elsevier, 2006, pp. 875-904.

[2] M. Zerola, J. Lauret, R. Barták, M. Šumbera, Using constraint programming to resolve multi-source / multi-site data movement paradigm on the Grid, in n *Advanced Computing and Analysis Techniques in Physics Research* PoS(ACAT08)039, 2008.

[3] A. Chabrier, Heuristic branch-and-price-and-cut to solve a network design problem, in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems CP-AI-OR 03*, Montreal, Canada, May 2003.

[4] C. Frei and B. Faltings, Resource allocation in networks using abstraction and constraint satisfaction techniques, in *Principles and Practice of Constraint Programming - CP 1999*, Alexandria, Virginia, October 1999.

[5] M. Zerola, M. Šumbera, R. Barták, J. Lauret, Using Constraint Programming to Plan Efficient Data Movement on the Grid, in *Proceedings of the 21st IEEE International Conference on Tools with Artificial Intelligence*, IEEE Computer Society, 2009, pp. 729-733

[6] L. Ros, T. Creemers, E. Tourouta, and J. Riera, A global constraint model for integrated routing and scheduling on a transmission network, in *7th International Conference on Information Networks, Systems and Technologies*, Minsk, October 2001.

[7] N. Beldiceanu and E. Contejean, Introducing global constraints in CHIP, in *Mathematical and Computer Modelling*, 12:97–123, 1994.

[8] R. Barták, M. Zerola, S. Slušný, Towards Routing for Autonomous Robots: Using Constraint Programming in Anytime Path Planner, submitted to ICAART 2011

[9] J.C. Régin, Generalized Arc Consistency for Global Cardinality Constraint, in *Proceedings of AAAI*, AAAI Press, 1996, pp. 209-215.

[10] W. Ouaja and B. Richards. A hybrid solver for optimal routing of bandwidth guaranteed traffic. In *INOC2003*, 2003, pp. 441–447.

[11] P. Baptiste, C. Le Pape, Edge-finding constraint propagation algorithms for disjunctive and cumulative scheduling, in *Proceedings of the Fifteenth Workshop of the U.K. Planning Special Interest Group* (PLANSIG), 1996.

[12] S. Loudni, P. David, and P. Boizumault. On-line resource allocation for ATM networks with rerouting, in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems CP-AI-OR 03*, Montreal, Canada, May 2003.

[13] Y. Caseau, F. Laburthe, Disjunctive scheduling with task intervals, LIENS Technical Report 95-25, Laboratoire d'Informatique de l'Ecole Normale Superieure, 1995.

[14] P. Torres, P Lopez, On Not-First/Not-Last conditions in disjunctive scheduling, in *European Journal of Operational Research* 127, 2000, pp. 332-343.

[15] R. Dechter, *Constraint Processing*. Morgan Kaufmann, 2003.

[16] J.C. Régin, A filtering algorithm for constraints of difference in CSPs, in *Proceedings of AAAI*, AAAI Press, 1994, pp. 362-367.

[17] G. Pesant, A Regular Language Membership Constraint for Finite Sequences of Variables, in *Principles and Practice of Constraint Programming*, LNCS 3285, Springer, 2004, pp. 482-495.

[18] R. Barták, Modelling Resource Transitions in Constraint-Based Scheduling, in *Proceedings of SOFSEM*, LNCS 2540, Springer, 2002, pp. 186-194.

[19] C. Bessiere, E. Hebrard, B. Hnich, Y. Kiziltan, C.G. Quimper, T. Walsh, Reformulating global constraints: The Slide and Regular Constraints, in *Proceedings of SARA*, LNCS 4612, Springer, 2007, pp. 80-92.