Symmetry Breaking in a Rack Configuration Problem

Zeynep Kızıltan and Brahim Hnich
Computer Science Division
Department of Information Science
Uppsala University, Box 513, S – 751 20 Uppsala, Sweden
{Zeynep.Kiziltan, Brahim.Hnich}@dis.uu.se

Abstract

We address the issue of reducing symmetry in a rack configuration problem, which can be modelled as a constraint satisfaction problem. The problem has a large number of symmetries that are difficult to break. We propose considering different views of the problem, which leads to different models. Within these different models, we seek opportunities for reducing the symmetries. The best result was obtained by integrating a primal model of the problem with its corresponding dual model. Experimental results on the integrated model show an exponential decrease in terms of failures and run-time, compared to all the other models considered.

1 Introduction

Having symmetry in a constraint satisfaction problem (CSP) makes the solving task extremely difficult. Failing to recognize and exploit symmetries inflates the size of the search space. If several branches in the search space are related by a symmetry, then it is sufficient to visit only one of them. The authors in [Benhamou and Sais, 1992; Benhamou, 1994] study how to find and exploit symmetries in propositional encodings of CSPs. Smith [2001] discusses different approaches to reducing symmetry. The first is to remodel the problem, so that the new model has less symmetries (e.g., [Smith, 2001]). The second is to add symmetry-breaking constraints to the model, so that some symmetric solutions are eliminated (e.g., [Smith, 2001]). The third is to add constraints during search, so that some branches containing symmetric solutions are not explored (e.g., [Backofen and Will, 1999; Gent and Smith, 2000; Smith, 2001]).

In this paper, we address the issue of reducing symmetry in a rack configuration problem, which can be modelled as a CSP. The problem inherently has a large number of symmetries that are difficult to break. We here attempt reducing symmetries through remodelling, with the hope that some of the resulting models allow an easy formulation of more symmetry-breaking constraints, compared to the original model. We do not address the effect of using different variable and value ordering heuristics. Also, we use OPL as a platform to test our results.

An OPL model of this problem is given in [Van Hentenryck, 1999]. However, in this model, the added symmetry-breaking constraints do not remove all the symmetries. Therefore, we propose different models of the problem, with the hope that more symmetries can be precluded in a static way. Firstly, we design a primal and its corresponding dual model of the problem. The primal model has simple constraints, but suffers from a large number of symmetries. On the other hand, its corresponding dual model allows us to state an extra symmetry-breaking constraint, but suffers from complex constraints. Thus, towards reducing symmetries without trading off the efficiency, our approach is to carefully integrate the primal and the dual model. Experimental results on the integrated model show an exponential decrease in terms of failures and run-time, compared to all the other models considered.

The rest of this paper is organised as follows. In Section 2, we explain the problem, and introduce a model of the problem published in [Van Hentenryck, 1999]. In Section 3, we propose two different models, each of which is a dual of the other one. Then, in Section 4, we present an integration of the models proposed. An alternative model is examined in Section 5. Finally, in Section 6, we summarise our contribution, and point out our future directions of research.

2 The Problem and Initial Models

In this section, we first describe the rack configuration problem. Later, we explain the model of the problem published in [Van Hentenryck, 1999], as well as a variant of this model, which employs a different symmetry-breaking constraint.

2.1 Problem Description

The rack configuration problem [Van Hentenryck, 1999] consists of plugging a set of electronic cards into racks with electronic connectors. Each card belongs to a certain card type. A card type is characterised by the power it requires, and a demand, which designates how many cards of that type have to be plugged. In order to plug a card into a rack, the rack needs to use a rack model.

Each rack model is characterised by the maximal power it can supply, its number of connectors, and its price. A rack model can be used by any rack. Each card plugged into a rack uses a connector. The problem is to decide how many of the available racks are needed, and which rack models the

Rack Model	Power	Connectors	Price
1	150	8	150
2	200	16	200

Card Type	Power	Demand
1	20	10
2	40	4
3	50	2
4	75	1

Figure 1: Rack model and card type specifications.

Rack Model	Card Type			
	1	2	3	4
2	6	2	0	0
2	1	2	2	0
1	3	0	0	1

Figure 2: An optimal solution for Figure 1.

racks use in order to plug all the cards such that the total price is minimised. Figure 1 shows an instance of this problem, provided that the number of available racks is 5. An optimal solution of this instance is shown in Figure 2. In this solution, only 3 racks are needed. One of the racks uses rack model 1, while two of the racks use rack model 2. The rack using rack model 1, for example, has 3 cards of type 1, and 1 card of type 4.

The constraints of the problem and the cost function can be summarised as the following:

- The *connector-capacity constraint*: the number of cards plugged into a rack r must not exceed the number of connectors of the rack model used by r.
- The *power-capacity constraint*: the total power of the cards plugged into a rack r must not exceed the power of the rack model used by r.
- The *demand constraint*: all the cards have to be plugged into some rack.
- The *cost function* is the sum of the prices of the rack models used by all the racks needed.

2.2 Initial Models

The main idea of the model presented in [Van Hentenryck, 1999] is to assign a rack model to every available rack. Since some of the racks might not be needed in an optimal solution, a dummy rack model is introduced (i.e., a rack is assigned the dummy rack model when the rack is not needed). Furthermore, for every available rack, the number of cards — of a particular card type — plugged into the rack has to be determined.

Data Representation

The model first defines the following sets:

- $RackModels = \{0, \ldots, NbRackModels - 1\}$, where $NbRackModels$ is the given number of rack models, including the dummy rack model.
- $CardTypes = \{0, \ldots, NbCardTypes - 1\}$, where $NbCardTypes$ is the given number of card types.
- $Racks = \{0, \ldots, NbRacks - 1\}$, where $NbRacks$ is the given number of available racks.

The model then defines the dummy rack model as a rack model where the maximal power it can supply, its number of connectors, and its price are all set to 0.

The mapping from racks to rack models is represented by an array of variables R indexed by $Racks$. The array R ranges over $RackModels$. In order to represent the number of cards – of a particular card type – plugged into a particular rack, a 2D array of variables C, indexed by $Racks$ and $CardTypes$, is introduced. The array C ranges over $\{0, \ldots, max\}$, where max is the maximum number of cards that can be plugged into any rack (using any rack model).

Problem Constraints and Cost Function

In this model, the constraints of the problem are stated as follows:

- The *connector-capacity constraint*:

  $$\forall r \in \mathit{Racks} \,. \sum_{c \in CardTypes} C[r,c] \;\leq\; \mathit{connector}(R[r])$$

  where *connector* is a function returning the number of connectors of a rack model.
- The *power-capacity constraint*:

  $$\forall r \in \mathit{Racks} \,. \, \textstyle\sum_{c \in CardTypes} C[r,c] * \mathit{power}_1(c) \;\leq\; \mathit{power}_2(R[r])$$

  where $power_1$ and $power_2$ are functions returning the power required for a card type, and the maximal power of a rack model, respectively.
- The *demand constraint*:

  $$\forall c \in CardTypes \,. \sum_{r \in Racks} C[r,c] \;=\; \mathit{quantity}(c)$$

  where $quantity$ is a function returning the quantity required for a card type.
- The *cost function*:

  $$\mathit{Cost} = \sum_{r \in Racks} \mathit{price}(R[r])$$

  where $price$ is a function returning the price of a rack model.

Symmetry Breaking Constraints

Even though the constraints stated so far correctly capture the problem constraints, the model suffers from the symmetries in the problem. Firstly, any permutation of the array R also is a solution. Consequently, the constraint:

$$\forall r \in \{1, \ldots, NbRacks - 1\} \,.\; R[r-1] \geq R[r]$$

forces a non-increasing ordering on the array R, which removes this symmetry. Secondly, any two adjacent racks using the same rack model are equivalent, and any permutation of the cards plugged into these racks also is a solution. It is very difficult to break this kind of symmetry. However, one can reduce *some* of such symmetries by stating that whenever two adjacent racks are assigned the same rack model, one of them must have at least as many cards as the other:

$$\forall r \in \{1,\ldots,NbRacks-1\}\,.\ R[r-1]=R[r] \rightarrow \sum_{c_1 \in CardTypes} C[r-1,c_1] \ \geq\ \sum_{c_2 \in CardTypes} C[r,c_2]$$

We refer to this model as model A. In [Van Hentenryck, 1999], however, they force a non-increasing ordering just on the cards of type 0, which is stated as follows:

$$\forall r \in \{1,\ldots,NbRacks-1\}\,.\ R[r-1]=R[r] \ \rightarrow C[r-1,0] \geq C[r,0]$$

We refer to this model as model B.

As also noted in [Van Hentenryck, 1999], the added constraints do *not* remove *all* the symmetries. On the other hand, the remaining symmetries are harder to prevent in a static way in this model. This brings about the need of an investigation of alternative models that *can* reduce the symmetries.

3 Alternative Models

In this section, we firstly propose two different models of the problem. Then, we compare their performance to the performance of the models described in Section 2, using three different instances.[1]

3.1 Primal Model

Note that the given number of available racks ($NbRacks$) designates the maximum number of racks that can be needed. Moreover, a rack uses a rack model in order to plug cards into it. Furthermore, a rack model can be used by any rack. Hence, a particular rack model can be used at most $NbRacks$ times. Additionally, the number of cards — of a specific card type — to be plugged is known. This leads us to view the problem as a relation between every possible use of each rack model and the cards. In this modelling, a dummy rack model is not needed.

Data Representation

The model keeps track of the same sets defined in Model A, with the exception that the set $RackModels$ now excludes the dummy rack model. Additionally, the following sets are defined:

- $P = \{\langle m,r\rangle \mid m \in RackModels \wedge r \in Racks\}$
- $Q = \{\langle c,t\rangle \mid c \in CardTypes \wedge t \in \{1,\ldots,quantity(c)\}\}$

Every tuple $\langle m,r\rangle \in P$ corresponds to the r^{th} use of rack model m, and every tuple $\langle c,t\rangle \in Q$ corresponds to the t^{th} card of card type c.

The relation between every possible use of each rack model and the cards is represented by a 2D array of variables Rel, indexed by P and Q. The array Rel ranges over $\{0,1\}$, where $Rel[\langle m,r\rangle,\langle c,t\rangle] = 1$ iff the t^{th} card of the card type c is related to the r^{th} use of the rack model m.

On the other hand, the problem states that $NbRacks$ racks are available. As a consequence, the relation is actually between a subset of P, and Q, where the size of the subset must be restricted not to exceed $NbRacks$ (we call this the *cardinality constraint*). The model captures this by introducing a set variable W, represented by an array of variables, indexed by P, and ranging over $\{0,1\}$.[2] This array represents a subset of P, where $W[\langle m,r\rangle] = 1$ iff the tuple $\langle m,r\rangle$ is in W. Also, this tuple $\langle m,r\rangle$ must be related to at least one card (we call this the *linking constraint*).

Problem Constraints and Cost Function

In this model, the constraints of the problem can be stated as follows:

- The *connector-capacity constraint*:

  $$\forall\langle m,r\rangle \in P\,.\ \sum_{\langle c,t\rangle \in Q} Rel[\langle m,r\rangle,\langle c,t\rangle] \ \leq\ connector(m)$$

  guarantees that the number of cards in Q related to a tuple $\langle m,r\rangle$ in P does not exceed the number of connectors of the rack model m, where *connector* is a function returning the number of connectors of a rack model.

- The *power-capacity constraint*:

  $$\forall\langle m,r\rangle \in P\,.\ \sum_{\langle c,t\rangle \in Q} Rel[\langle m,r\rangle,\langle c,t\rangle] * power_1(c) \ \leq\ power_2(m)$$

  guarantees that the total power of the cards in Q related to a tuple $\langle m,r\rangle$ in P does not exceed the maximal power of the rack model m, where $power_1$ and $power_2$ are functions returning the power required for a card, and the maximal power of a rack model, respectively.

- The *demand constraint*:

  $$\forall\langle c,t\rangle \in Q\,.\ \sum_{\langle m,r\rangle \in P} Rel[\langle m,r\rangle,\langle c,t\rangle] \ =\ 1$$

  ensures that each card in Q is related to exactly one tuple in P.

- The *cardinality constraint*:

  $$\sum_{\langle m,r\rangle \in P} W[\langle m,r\rangle] \ \leq\ NbRacks$$

  restricts the maximum number of tuples in W to $NbRacks$.

- The *linking constraint:*

  $$\forall\langle c,t\rangle \in Q\,.\,\forall\langle m,r\rangle \in P\,.\ Rel[\langle m,r\rangle,\langle c,t\rangle] = 1 \rightarrow W[\langle m,r\rangle] = 1$$

  states that if a card in Q is related to a tuple $\langle m,r\rangle$ in P, then the tuple $\langle m,r\rangle$ must exist in the subset W.

- The *cost function*:

  $$Cost = \sum_{\langle m,r\rangle \in P} W[\langle m,r\rangle] * price(m)$$

  defines the cost as the sum of the prices of all tuples in W, where *price* is a function returning the price of a rack model.

[1] All models have been implemented in OPL [Van Hentenryck, 1999]. Its default labelling strategy is the smallest-domain-first variable ordering heuristic.

[2] OPL does not support set variables, unlike ILOG SOLVER.

Symmetry Breaking Constraints
Regretfully, this model contains many symmetries. In particular, any permutation of the different uses of a particular rack model also is a solution. By posing the constraint:

$$\forall m \in RackModels \,.\, \forall r \in \{1, \ldots, NbRacks - 1\} \,.\\ W[\langle m, r-1\rangle] \;\geq\; W[\langle m, r\rangle]$$

we give a non-increasing ordering on the different uses of each rack model, and thus remove this symmetry. Moreover, whenever the same rack model m is used by two adjacent racks r_1 and r_2, any permutation of the related cards of $\langle m, r_1\rangle$ and $\langle m, r_2\rangle$ also leads to a new solution. By stating:

$$\forall m \in RackModels \,.\, \forall r \in \{1, \ldots, NbRacks - 1\} \,.\\ W[\langle m, r-1\rangle] = 1 \;\wedge\; W[\langle m, r\rangle] = 1 \rightarrow\\ \sum_{\langle c,t\rangle \in Q} Rel[\langle m, r-1\rangle, \langle c, t\rangle] \;\geq\\ \sum_{\langle c,t\rangle \in Q} Rel[\langle m, r\rangle, \langle c, t\rangle]$$

we impose a non-increasing ordering on the numbers of their related cards. Note that this constraint does *not* remove *all* such symmetries, but it is difficult to totally remove such symmetries in this model.

Furthermore, any permutation of the tuples in P related to all cards of a certain card type in Q also is a solution. Unfortunately, this symmetry is very difficult be break in this model.

3.2 Dual Model

In the primal model, the problem is viewed as a relation between a subset of every possible use of each rack model and the cards. As the *demand constraint* suggests, every card in Q must be related to exactly one tuple in P. This hints that the dual representation of the primal model is a mapping from the cards to a subset of every possible use of each rack model.

Data Representation
The model maintains the same sets as the primal model. The mapping from the cards to a subset of every possible use of each rack model is represented by using an array of variables Map. The array Map is indexed by Q, and ranges over P. The mapping of the t^{th} card of a card type c to the r^{th} use of a rack model m is thus captured whenever $Map[\langle c, t\rangle] = \langle m, r\rangle$. Similarly to the primal model, a set variable W is introduced to represent a subset of P.

Problem Constraints and Cost Function
Since the dual model represents a subset of P using the array W, like the primal model, the *cardinality constraint* on W and the *cost function* in this model are identical to the ones in the primal model. The mapping from Q to P makes sure that each card in Q is mapped to exactly one tuple in P. Thus, the *demand constraint* is fully captured by the data modelling. The formulation of the remaining constraints in the dual model can be the following:

- The *connector-capacity constraint*:

  $$\forall \langle m, r\rangle \in P \,.\\ card(\{\langle c, t\rangle \in Q \mid Map[\langle c, t\rangle] = \langle m, r\rangle\}) \;\leq\\ connector(m)$$

  guarantees that the number of cards in Q mapped to a tuple $\langle m, r\rangle$ in P does not exceed the number of connectors of the rack model m, where $card$ computes the cardinality of a set.
- The *power-capacity constraint*:

  $$\forall \langle m, r\rangle \in P \,.\\ \sum_{\langle c,t\rangle \in Q} (Map[\langle c, t\rangle] = \langle m, r\rangle) * power_1(c) \;\leq\\ power_2(m)$$

  guarantees that the total power of the cards in Q mapped to a tuple $\langle m, r\rangle$ in P does not exceed the maximal power of the rack model m.
- The *linking constraint*

  $$\forall \langle c, t\rangle \in Q \,.\, \forall \langle m, r\rangle \in P \,.\\ Map[\langle c, t\rangle] = \langle m, r\rangle \;\rightarrow\; W[\langle m, r\rangle] = 1$$

  states that if a card in Q is mapped to a tuple $\langle m, r\rangle$ in P, then the tuple $\langle m, r\rangle$ must exist in the subset W.

Note that the *connector-capacity* and the *power-capacity* constraints in this model get compiled into more complex constraints compared to the ones in the primal model. For instance, the *power-capacity* constraint, when re-written as finite domain constraints, introduces Boolean variables that are reified with every equality statement $Map[\langle c, t\rangle] = \langle m, r\rangle$, and poses the summation constraints on those Boolean variables.

Symmetry Breaking Constraints
The symmetry on the different uses of a particular rack model exists also in this model, and the constraint to remove such a symmetry is identical to the one in the primal model. Moreover, whenever the same rack model m is used by two adjacent racks r_1 and r_2, any permutation of the cards mapping to $\langle m, r_1\rangle$ and $\langle m, r_2\rangle$ also leads to a new solution. By stating:

$$\forall m \in RackModels \,.\, \forall r \in \{1, \ldots, NbRacks - 1\} \,.\\ W[\langle m, r-1\rangle] = 1 \;\wedge\; W[\langle m, r\rangle] = 1 \rightarrow\\ card(\{\langle c, t\rangle \in Q \mid Map[\langle c, t\rangle] = \langle m, r-1\rangle\}) \leq\\ card(\{\langle c, t\rangle \in Q \mid Map[\langle c, t\rangle] = \langle m, r\rangle\})$$

we impose a non-increasing ordering on the numbers of cards they are mapped to. Note that this constraint does *not* remove *all* such symmetries, but it is difficult to totally remove such symmetries in this model. Also, this constraint is more complex than the corresponding one in the primal model.

Furthermore, any permutation of the tuples in P mapped by all cards of a certain card type in Q also is a solution. This symmetry can easily be broken in this model by enforcing an ordering on all cards of a certain card type in Q:

$$\forall c \in CardTypes \,.\, \forall t \in \{2, \ldots, quantity(c)\} \,.\\ Map[\langle c, t-1\rangle] \;\geq\; Map[\langle c, t\rangle]$$

Note that we could not break this symmetry in the primal model.

3.3 Experimental Results

In order to test the performance of each model proposed so far, we used 3 problem instances. The first instance is the

Models	Ins. 1 fails	time	Ins. 2 fails	time	Ins. 3 fails	time
A	49	$0.02s$	$\approx 182K$	$60s$	$\approx 164M$	$14.06h$
B	47	$0.01s$	$\approx 64K$	$14.65s$	$\approx 112M$	$7.5h$
Primal	46	$0.24s$	$\approx 5M$	$4h$	$\approx 15M$	$11.3h$
Dual	482	$0.82s$	$\approx 5M$	$8.3h$	$\approx 9M$	$16h$

Table 1: The effect of remodelling

Models	Ins. 1 fails	time	Ins. 2 fails	time	Ins. 3 fails	time
A	49	$0.02s$	$\approx 182K$	$60s$	$\approx 164M$	$14.06h$
B	47	$0.01s$	$\approx 64K$	$14.65s$	$\approx 112M$	$7.5h$
Primal	46	$0.24s$	$\approx 5M$	$4h$	$\approx 15M$	$11.3h$
Dual	482	$0.82s$	$\approx 5M$	$8.3h$	$\approx 9M$	$16h$
Integrated	27	$0.34s$	82	$3.7s$	$\approx 5K$	$45.76s$

Table 2: The effect of integration of primal and dual models

one provided in [Van Hentenryck, 1999] and given in Figure 1, whereas we hand-crafted the other two instances as we failed to find real-life problem instances in the literature. The problem size and the number of symmetries increase from instance 1 to instance 2, as well as from instance 2 to instance 3. Our goal is to monitor the performance of each model with respect to the reducing of symmetries. Thus, we recorded the number of failures achieved by every model, as well as the run-times (as some models may reduce the symmetries without pay-off in terms of runtime). The results are shown in Table 1. Note that s stands for seconds, h for hours, K denotes a thousand, and M denotes a million.

As seen in Table 1, model B performs better than model A on all instances. This may be explained by the fact that the second symmetry-breaking constraint of model A is more costly. The primal model spends always more time than model B. However, on the third instance, the primal model performs less failures, compared to model B. The dual model has the worst run-times, even though it achieves the least number of failures on the third instance. As a matter of fact, all the models degenerate in terms of run-time, as the problem size and the number of symmetries increase.

We observe that the primal model has simpler constraints, compared to the dual model. That might be the reason why the primal model is much more efficient than the dual model in terms of run-time. In principle, the dual model poses more symmetry-breaking constraints, compared to the primal model. In practice, however, it becomes beneficial only when the gain from reducing the search space surpasses the overhead. So far, we observe that the dual model has the potential of reducing more symmetries, as the problem size and the number of symmetries increase. Unfortunately, as seen in Table 1, the overhead is still high. We conjecture that the overhead comes mainly from the complex constraints of the dual model, rather than the extra symmetry-breaking constraint. It would thus be nice to benefit from the efficiency of the primal model due to its simple constraints, as well as from the capability of the dual model in reducing the search space due to its extra symmetry-breaking constraint. This suggests that integrating the primal model with the dual model, in a clever way, might overcome the cons of the primal with the pros of the dual model, and vice versa.

4 Integrating the Models

The primal and dual models can be combined so as to obtain a new model of the same problem. Integration of different models of a problem was previously studied by Cheng *et al.* [1999] and Smith [2000; 2001]. By integrating different models, the domain pruning carried out in each model may be improved. Such an integration is achieved by introducing constraints to link the variables of the two separate models. These linking constraints are called *channelling constraints* [Cheng *et al.*, 1999].

In our case, the integration of the primal with the dual model introduces the following channelling constraint:

$$\forall \langle c,t\rangle \in Q \,.\, \forall \langle m,r\rangle \in P \,.$$
$$Rel[\langle m,r\rangle,\langle c,t\rangle] = 1 \;\leftrightarrow\; Map[\langle c,t\rangle] = \langle m,r\rangle$$

which states that a tuple $\langle m,r\rangle$ in P is related to a card $\langle c,t\rangle$ in Q if and only if the card $\langle c,t\rangle$ in Q is mapped to the tuple $\langle m,r\rangle$ in P.

In [Cheng *et al.*, 1999], the integration is performed by linking the two *complete* models of the same problem, using channelling constraints. However, Smith in [2000] shows that it is cumbersome to include the constraints of each model. Maintaining the two sets of variables, linking them using the channelling constraints, and dropping the constraints of either model leads to a minimal integrated model. The experimental results in [Smith, 2000] and the theoretical work in [Walsh, 2001] on permutation problems show that the complete integrated model achieves the same amount of pruning as the minimal integrated model does, but takes much a longer time.

In our case, the objective of integration is to overcome the deficiency of the constraints of the dual model by using the simpler and more efficient constraints of the primal model. Additionally, we aim at taking the advantage of the effect of the extra symmetry-breaking constraint in the dual model. Note that the primal variables are Rel and W, and the dual variables are Map and W, with W being a shared variable between both models. Hence, it becomes obvious to integrate our models by stating the constraints of the problem on only Rel and W, while stating just the extra symmetry-breaking constraint on Map. Consequently, the channelling constraint triggers the constraint propagation performed in one model to the other one. Finally, even though we do not address the effect of variable and value ordering heuristics, we had to make a choice in the integrated model on which variables to branch on, as we do not need to label all the variables. Since the primal model has more variables, it was just natural to branch on the dual variables.

The significance of the integration can easily be seen in Table 2. The integrated model achieves the least number of failures on all instances. The difference grows dramatically as the problem size and the number of symmetries increase. In terms of run-time, we observe that the integration does not pay off on the first instance, but has a dramatic effect on the other instances. In fact, the integrated model has the lowest run-times on the last two instances. Hence, the integration seriously benefits from each of the models, both in terms of run-time and in terms of reducing the search space.

5 A Model Based on Set Variables

Since the primal model introduced many Boolean variables, it would be nice to seek an alternative model with less variables but still with efficient way of posing the problem constraints. This model could then be integrated with the dual one. We here explore whether set variables can be used to achieve that.

In fact, every possible use of a rack model ($\langle m, r\rangle$) is related to either some cards or none at all. This suggests that every possible use of a rack model can be viewed as a set variable having domain the set of cards Q. Thus, we can introduce a set S of set variables:

$$\begin{aligned} S = \quad & \{S_1^1, \ldots, S_1^{NbRacks}, \ldots, \\ & S_{NbModels}^1, \ldots, S_{NbModels}^{NbRacks}\} \end{aligned}$$

where every S_m^r in S represents the r^{th} use of rack model m. Our objective now is to find a subset of S, say T, whose cardinality does not exceed $NbRacks$. In other words, the set of cards Q is partitioned into the elements of T.

We implemented this model in CONJUNTO [Gervet, 1997], which is a finite-set constraint solver in ECLIPSE. Stating the partitioning conditions and the problem constraints was straightforward by using the set variables and the (set) constraint predicates provided by the language. Unfortunately, CONJUNTO does not allow us to declare a set variable whose domain is a set of set variables. Namely, the declaration $T \subseteq S$ is not possible in CONJUNTO. Thus, we could not represent T as a set variable, and we were forced to introduce Boolean variables to pose the *cardinality constraint*, and to state the *cost function*. Even though CONJUNTO provides some high-level set constraint predicates, which indeed helped us to state the constraints of this set partitioning problem, it fails to enable us to express the *cardinality constraint* and the *cost function* in an elegant way.

Moreover, in CONJUNTO, the logical constraint predicates (e.g., $\leftrightarrow$) cannot be used together with set constraint predicates (e.g., $\in$). Thus, we are currently unable to state the channelling constraints. However, we believe that this integration of the set variable model and the dual model is very promising, and further investigation has to be done.

6 Conclusion

In this paper, we considered a rack configuration problem that has many symmetries that are difficult to break. After introducing models A and B, we proposed a primal model and its dual model of the problem. The primal model has simple constraints, but suffers from a large number of symmetries. On the other hand, its dual model allows us to state an extra symmetry-breaking constraint, but suffers from complex constraints. Thus, towards reducing symmetries without trading off the efficiency, our approach was to carefully integrate the primal model with its dual model. The integrated model states the constraints of the problem only on the primal variables, while only the extra symmetry-breaking constraint is stated on the dual variables. The integrated model uses the channelling constraints to trigger the constraint propagation performed in one model to the other one. Experimental results on the integrated model show a significant decrease in terms of failures and run-time, compared to all the other models considered.

Our next steps will be to test our integrated model on real-life problem instances. Also, the integrated model makes it possible to branch on either (primal or dual) set of variables, which opens the door to a cheap implementation of value ordering heuristics. Furthermore, the effect of branching on both sets of variables of the integrated model should be studied. Additionally, further investigation is needed to be able to add more constraints during search, as the integrated model still cannot preclude all the symmetries. Finally, we will consider the integration of the set variable model with the dual model.

Acknowledgements. This research is partly funded under grant 221-99-369 of TFR, the Swedish Research Council for Engineering Sciences. We would also like to acknowledge our advisor Pierre Flener for his valuable comments.

References

[Backofen and Will, 1999] R. Backofen and S. Will. Excluding symmetries in constrained-based search. In: J. Jaffar (ed), *Proc. of CP'99*, pp. 73–87. Lecture Notes in Computer Science, volume 1713. Springer-Verlag, 1999.

[Benhamou, 1994] B. Benhamou. Study of symmetry in constraint satisfaction problems. In: A. Borning (ed), *Proc. of PPCP'94*, pp. 246–254. Lecture Notes in Computer Science, volume 874. Springer-Verlag, 1994.

[Benhamou and Sais, 1992] B. Benhamou and L. Sais. Theoretical study of symmetries in propositional calculus and applications. In: D. Kapur (ed), *Proc. of CADE-11*, pp. 281–294. Lecture Notes in Artificial Intelligence, volume 607. Springer-Verlag, 1992.

[Cheng *et al.*, 1999] B.M.W. Cheng, K.M.F. Choi, J.H.M. Lee, and J.C.K. Wu. Increasing constraint propagation by redundant modelling: An experience report. *Constraints*, 4:167–192, 1999.

[Gent and Smith, 2000] I.P. Gent and B. Smith. Symmetry breaking during search in constraint programming. In: W. Horn (ed), *Proc. of ECAI'2000*, pp. 599–603. IOS Press, 2000.

[Gervet, 1997] C. Gervet. Interval propagation to reason about sets: Definition and implementation of a practical language. *Constraints* 1(3):191–244, 1997.

[Smith, 2000] B.M. Smith. Modelling a permutation problem. RR 18, University of Leeds (UK), School of Computer Studies, 2000.

[Smith, 2001] B.M. Smith. Reducing symmetry in a combinatorial design problem. RR 01, University of Leeds (UK), School of Computer Studies, 2001.

[Van Hentenryck, 1999] P. Van Hentenryck. *The* OPL *Optimization Programming Language*. The MIT Press, 1999.

[Walsh, 2001] T. Walsh. Permutation problems and channelling constraints. TR 26, APES Group, 2001.