Stronger integration of circuit and alldifferent propagators for the Hamiltonian Cycle Problem

Alessandro Bertagnon[1], Marco Gavanelli[2]

[1]Department of Environmental and Prevention Sciences, University of Ferrara, C.so Ercole I D'Este, 32, Ferrara, Italy
[2]Department of Engineering, University of Ferrara, Via Saragat 1, Ferrara, Italy

Abstract

Global constraints are one of the features that make Constraint Programming an effective solution scheme. The first global constraint, named alldifferent, is also one of the most used. In order to solve in Constraint Programming routing problems, such as the Hamiltonian Circuit Problem, the Travelling Salesperson Problem and many of their variants, an effective solution is to use a constraint model containing alldifferent and the circuit constraint, necessary to avoid sub-circuits.

In this paper, we propose a combination of alldifferent and circuit that reuses the data structures of the alldifferent constraint to perform further propagation for the circuit constraint. The new combination introduces negligible overhead, and experimental results show that it can be effective when solving the Hamiltonian Circuit Problem.

Keywords

Constraint Logic Programming on Finite Domains, Hamiltonian Circuit Problem, Global constraints, alldifferent constraint, circuit constraint

1. Introduction

In the field of Constraint Programming (CP), global constraints such as `alldifferent` and `circuit` play a crucial role in reducing the search space and improving the efficiency while solving a wide range of Constraint Optimization Problems (COPs). These constraints are particularly relevant in complex applications like scheduling, timetabling, vehicle routing, and problems in graph theory.

The `alldifferent` constraint is essential for ensuring that all variables within a set assume distinct values. The `circuit` constraint enforces the existence of a Hamiltonian cycle within a graph.

Despite their practical utility, these constraints pose significant computational challenges, especially when combined in problems such as the Hamiltonian Cycle Problem (HCP), a well-known NP-complete problem. Achieving efficient constraint propagation while maintaining computational tractability of the propagation is a core issue in CP.

While significant advances have been made in the propagation techniques for both the `alldifferent` and `circuit` constraints individually, much less attention has been devoted to the potential synergies between these constraints when combined. For example, in problems like the HCP, both constraints are naturally present: the `alldifferent` constraint is used to ensure that each node in the cycle has a unique successor, while the `circuit` constraint ensures that these successors form a valid cycle without subtours.

In this paper, we propose a novel approach that integrates `alldifferent` and `circuit` constraints more deeply, with a focus on achieving better pruning during propagation, which in turn leads to improved overall performance. Our approach is motivated by the observation that by leveraging the underlying structure of Hall sets, typically used in the propagation of `alldifferent`, we can obtain additional pruning in the context of `circuit` constraint.

AI4CC-IPS-RCRA-SPIRIT 2024: International Workshop on Artificial Intelligence for Climate Change, Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy. November 25-28th, 2024, Bolzano, Italy [1].
alessandro.bertagnon@unife.it (A. Bertagnon); marco.gavanelli@unife.it (M. Gavanelli)
0000-0003-2390-0629 (A. Bertagnon); 0000-0001-7433-5899 (M. Gavanelli)

We provide a theoretical overview that analyzes the computational complexity of the integrated approach. Although it is well-known that the combination of `alldifferent` and `circuit` constraints generates an NP-complete problem, we show a combined propagation that is polynomial and, although it does not achieve Generalized Arc-Consistency, it provides an improvement in the solution time.

Finally, we provide an extensive experimental evaluation, comparing our integrated approach against existing methods on a variety of problem instances of the HCP.

2. Preliminaries and notation

Definition 2.1. *(Constraint Satisfaction Problem) A Constraint Satisfaction Problem (CSP) is a triple* $\mathscr{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ *where* $\mathcal{X}$ *is a set of decision variables* $\{x_1, x_2, \ldots, x_n\}$, $\mathcal{D}$ *is a set of domains* $\{D_1, D_2, \ldots, D_n\}$ *and* $\mathcal{C}$ *a set of constraints* $\{c_1, c_2, \ldots, c_m\}$.

Each domain D_i is the set of all possible values that can be assigned to the variable x_i. Each constraint c_i consists of a pair $\langle R_i, S_i \rangle$ where R_i is a relation between the variables S_i participating in the constraint. The set S_i is called the *scope* of R_i. R_i results in a subset of the cartesian product of the domain of the variables in S_i.

Let $G = (V, E)$ be a graph, where V is a set of nodes and E is a set of edges. A *path* in G is a sequence $p_{v_{s_0}\text{-}v_{s_k}} = v_{s_0} e_{s_0,s_1} v_{s_1} \ldots e_{s_{k-1},s_k} v_{s_k}$ such that

1. $v_{s_0}, v_{s_1}, \ldots, v_{s_k} \in V$ and are all distinct, and
2. $e_{s_0,s_1}, \ldots, e_{s_{k-1},s_k} \in E$.

Since a path is uniquely identified by the sequence of its nodes (or of its edges) in the proper order, to simplify the notation we will often write paths as sequences of nodes. Given a path $p_{v_{s_0}\text{-}v_{s_k}}$, the sequence obtained by appending e_{s_k,s_0} to a path $p_{v_{s_0}\text{-}v_{s_k}}$ is also called a *circuit* c.

Definition 2.2. *Given a graph* G*, the HCP is the problem of finding a cycle in* G *that passes through all nodes, without taking twice the same edge.*

2.1. State of the Art for `alldifferent` and `circuit` constraints

The `alldifferent` constraint is one of the most used constraints in Constraint Logic Programming (CLP), and was subject of several works [2, 3, 4, 5, 6]. The works propose different tradeoffs between the pruning power (stronger consistency of the propagation) and the computational complexity for achieving it; a survey on the `alldifferent` constraint was published by Van Hoeve [7].

The first work on the `alldifferent` constraint [2] exploited graph-matching algorithms, and achieved the strongest possible level of consistency for a (single) constraint, namely (Generalized) Arc-Consistency.

Definition 2.3 (Generalized Arc-Consistency)**.** *A constraint* $c(x_1, \ldots, x_n)$ *is* Generalized Arc-Consistent *(or Hyper-Arc Consistent) if for each variable* $x_i \in \{x_1, \ldots, x_n\}$ *and for each value* $d_j \in D_{x_i}$ *there exist values*

$$d_1 \in D_{x_1}, \ldots, d_{i-1} \in D_{x_{i-1}}, d_{i+1} \in D_{x_{i+1}}, \ldots, d_n \in D_{x_n}$$

such that $(d_1, \ldots, d_{i-1}, d_j, d_{i+1}, \ldots, d_n) \in c$.

The complexity [2] was $O(n^{2.5})$, where n is the number of variables. Further improvements [6] on this version changed the implementation, but retaining the same computational complexity.

Leconte [8] proposed a faster propagation scheme, with complexity $O(n^2)$, which achieved a lower-level of consistency, named *range-consistency.* As for Arc-Consistency, the idea of range-consistency is that each element in the domain of each variable x_i should have supporting values in the domains of the other variables; however, in Range-Consistency the supporting values are sought in the minimal interval that encloses the domain, instead of the actual domain. This reduces the number of checks that are necessary to enforce such level of consistency, as it is no longer necessary to check all values in the domain, but only the extremes are considered. The downside is that Arc-Consistency can detect inconsistency in more instances.

Definition 2.4 (Range-Consistency). *A constraint $c(x_1, \ldots, x_n)$ is* Generalized Range-Consistent *if for each variable $x_i \in \{x_1, \ldots, x_n\}$ and for each value $d_j \in D_{x_i}$ there exist values*

$$\begin{array}{l} d_1 \in [\min(D_{x_1}), \max(D_{x_1})], \ldots, d_{i-1} \in [\min(D_{x_{i-1}}), \max(D_{x_{i-1}})], \\ d_{i+1} \in [\min(D_{x_{i+1}}), \max(D_{x_{i+1}})], \ldots, d_n \in [\min(D_{x_n}), \max(D_{x_n})] \end{array}$$

such that $(d_1, \ldots, d_{i-1}, d_j, d_{i+1}, \ldots, d_n) \in c$.

Puget [3] proposed a propagation algorithm with $O(n \log n)$ complexity, that achieved an even weaker notion of consistency, named *Bound-Consistency*; in Bound-Consistency a support is sought only for the bounds, i.e. the extreme elements in the domain of each variable:

Definition 2.5 (Bound-Consistency). *A constraint $c(x_1, \ldots, x_n)$ is* Generalized Bound-Consistent *if for each variable $x_i \in \{x_1, \ldots, x_n\}$ and for each value $d_j \in \{\min(D_{x_i}), \max(D_{x_i})\}$ there exist values*

$$\begin{array}{l} d_1 \in [\min(D_{x_1}), \max(D_{x_1})], \ldots, d_{i-1} \in [\min(D_{x_{i-1}}), \max(D_{x_{i-1}})], \\ d_{i+1} \in [\min(D_{x_{i+1}}), \max(D_{x_{i+1}})], \ldots, d_n \in [\min(D_{x_n}), \max(D_{x_n})] \end{array}$$

such that $(d_1, \ldots, d_{i-1}, d_j, d_{i+1}, \ldots, d_n) \in c$.

Further works on this version [4, 5] were not able to improve on the computational complexity of the algorithm, but they were able to improve on its speed in practice.

All the current implementations of `alldifferent` achieving Range or Bound-Consistency are based on the Hall theorem [9]; before introducing it, we define the domain of a set of variables:

Definition 2.6 (Domain of a set of variables). *Given a set of variables S, for each $x \in S$ let D_x be the domain of variable x. We indicate with $\mathcal{D}_S = \bigcup_{x \in S} D_x$.*

Clearly, if there is a set of variables K such that the number of variables in K is higher than the number of available values $|\mathcal{D}_K|$ for those variables, the `alldifferent` constraint is unsatisfiable; the following theorem states that also the vice-versa holds:

Theorem 2.1 ([3], based on [9]). *The constraint* `alldifferent`$([X_1, \ldots, X_n])$ *has solutions iff for each $K \subseteq \{X_1, \ldots, X_n\}$, $|K| \leq |\mathcal{D}_K|$.*

The implementations of `alldifferent` based on Range or Bound-Consistency differ mainly on the method used to find *Hall intervals*:

Definition 2.7 (Hall interval [3]). *Given a constraint* `alldifferent`$([X_1, \ldots, X_n])$ *and an interval I, let $vars(I)$ be the set of variables X_i such that $D_{X_i} \subseteq I$. We say that I is a* Hall interval *iff $|vars(I)| = |I|$.*

Clearly, if there is a Hall interval I, then the set of variables $vars(I)$ absorbs all the values of $\mathcal{D}_I$, meaning that all values in $\mathcal{D}_I$ can be safely removed from the domains of all other variables. This is exactly the propagation used by the algorithms based on Range or Bound-Consistency for the `alldifferent` constraint.

Also the algorithms that achieve Arc-Consistency [2, 6] identify sets of values I whose cardinality coincides with that of the set $vars(I)$, however, such sets are not necessarily intervals; we will name them *Hall sets*.

One of the most successful constraint models for the HCP is the *successor representation.* To each node x of the graph, a variable $Next_x$ is associated; its domain is the set of nodes that can be reached in one step from x. The intuitive meaning of this representation is that in a solution (i.e., in an Hamiltonian cycle), the successor of the the node x is the node assigned to $Next_x$.

The constraint model for the successor representation contains an `alldifferent` constraint on the set of $Next$ variables (since no two nodes can have the same successor in an Hamiltonian path) and a `circuit` constraint [10] on the $Next$ variables. The `circuit` constraint ensures that there are no sub-tours.

One simple implementation of the `circuit` constraint (and, indeed, the implementation implemented in the ECL^{i}PS^e Constraint Logic Programming language [11]) is based on the following reasoning: if in a partial assignment, a path P has already been assigned, starting from an initial node i and ending in a last node l, then $Next_l \neq i$, i.e. the successor of the last node l cannot be the initial node i. Clearly, this condition is valid only for partial assignments, not for complete ones (i.e., not for complete solutions), i.e. when the length of the partial path is strictly less than the number of nodes in the graph.

3. Related work

Caseau and Laburthe [12] propose a propagation technique for the `circuit` constraint through a simple and effective rule called `nocycle`. This rule is applied to prevent intermediate cycles during the solving of small Traveling Salesperson Problems (TSPs). Their method involves detecting paths of mandatory edges with lengths of at most $n - 1$ and eliminating the edge between the two endpoints of such paths to ensure circuit completeness. Additionally, they enhance the constraint-solving approach by utilizing assignment-based and spanning tree relaxations to filter out infeasible values, demonstrating how these techniques contribute to more effective propagation for the TSP and similar problems.

Kaya and Hooker [13] propose a new filtering approach to the `circuit` constraint based on separator graphs. Their method focuses on removing non-Hamiltonian edges by identifying and analyzing subgraphs using a vertex separator. To identify nonhamiltonian edges, Kaya and Hooker introduce a flow-based method that constructs capacitated flow graphs. These graphs are built for both out-degree and in-degree constraints, ensuring that each vertex in the Hamiltonian cycle has exactly one successor and one predecessor. If the flow on a given edge is zero and there is no augmenting path, that edge is nonhamiltonian and can be safely removed from the graph's domain. Their algorithm achieves a complexity of $O(|S|^5)$ for each separator S.

Francis and Stuckey [14] further explore various propagation techniques for the `circuit` constraint in the context of lazy clause generation solvers. They emphasize the importance of adding explanations and they studied its effect on the `circuit` constraint and its variants. The technique involves transforming each propagation step into a clause, which provides an explanation for domain reductions, helping the solver avoid previously encountered conflicts. Their research highlights the trade-off between the complexity of propagation algorithms and the reusability of explanations. While simpler algorithms generate smaller explanations, more powerful algorithms, such as Strongly Connected Components (SCC) based propagation, can yield significant performance gains by pruning the search space more effectively.

Isoart and Régin [15, 16] propose to improve the propagation of the Weighted Circuit Constraint (WCC) (a constraint tailored to solve the Travelling Salesperson Problem) by exploiting some properties of Hamiltonian graphs (i.e., graphs that admit an Hamiltonian circuit) based on finding k-cutsets. More precisely, a graph can be cut into two subgraphs by removing a set of edges, named a *cutset*; a cutset of cardinality k is called a k-cutset. In order for the graph to be Hamiltonian, there cannot exist a 1-cutset (also called a *bridge*) as there would be two separate parts connected only by one edge. Isoart and Régin develop efficient algorithm to detect cutsets of size up to three, and, reasoning on the cardinality of a cutset, are able to detect mandatory edges (edges that must necessarily belong to any Hamiltonian circuit) and edges that cannot belong to any Hamiltonian cycle.

4. Stronger interaction between `circuit` and `alldifferent`

Integrating the `circuit` and `alldifferent` constraints could unleash the possibility of further pruning. On the other hand, it is worth noting that the problem consisting only of the `alldifferent` and `circuit` constraints is known as the Hamiltonian Circuit problem, which is a well-known NP-complete problem [17]. But if a problem consisting of only one constraint is NP-complete, then also obtaining Generalized Arc-Consistency (GAC) of such a constraint is NP-complete [18]. This well-known result strongly reduces the hope to find a polynomial algorithm for GAC propagation of such a constraint. On the other hand, constraint propagation is executed in each node of the search space, and usually, in order for a propagation to be effective, a strong requirement is that it is achieved in polynomial time. For this reason, it is sensible to forego obtaining GAC of the `hamiltonian` constraint; this does not mean that effective pruning cannot be obtained for such a constraint: indeed a constraint is effective if the amount of pruning it performs (i.e., the reduction of the search space) compensates for the time spent in reasoning. E.g., one of the most successful constraints in CP is the `cumulative` constraint, for which there exist various implementations, none of which obtains GAC, since its cost would be NP-hard.

All the implementations of `alldifferent` amount to finding Hall sets efficiently, possibly trading speed for finding Hall sets with the number of Hall sets found.

Once these sets are known, it might be worthy to exploit them also for improving the propagation of the `circuit` constraint.

Theorem 4.1. *Let $H = \{h_1, \ldots, h_k\}$ be a set of nodes, $Next_H$ the corresponding variables in the successor representation, and $\mathcal{D}_{Next_H}$ the corresponding domain. If $\mathcal{D}_{Next_H} = H$ and $|H| < n$, then the HCP has no solution.*

Proof. Edges from the set H can only be connected to edges in the set $\mathcal{D}_H$, which coincides with H, so the set H is isolated. As $|H| < n$, the set H does not contain all nodes in the graph, so there are nodes that are unreachable from H. □

Note that a set satisfying Theorem 4.1 is, by definition, a Hall set.

Thus, it makes sense to exploit the efficient techniques developed in the literature to find Hall sets to get also additional pruning for the `circuit` constraint. Moreover, those same techniques used to find Hall sets are already embedded in the propagation algorithm of the `alldifferent` constraint, that is usually employed together with `circuit` in the same constraint model. Stated otherwise, since the `alldifferent` constraint already needs to search for Hall sets, it makes sense to get additional pruning, due to the need to remove sub-circuits.

Once a Hall set satisfying the condition of Theorem 4.1 is found, a proof is obtained that the current branch of the search tree does not lead to any solution, so a failure is raised. While failing early can save a lot of computation time, the CLP on Finite Domains (CLP(FD)) philosophy encourages to delete inconsistent values from the domains in order to focus the search on the promising parts of the search tree.

The following theorem provides a mean for eliminating values from domains before a failure.

Of course, the starting point of a circuit is unimportant.

Theorem 4.2. *Let $H = \{h_1, \ldots, h_k\}$ be a set of nodes with cardinality $k < n$ such that $\mathcal{D}_{Next_H}$ (the domain of the corresponding variables) is a Hall set. Assume that $|H \setminus \mathcal{D}_{Next_H}| = 1$; let h_s be the only element in $I = H \setminus \mathcal{D}_{Next_H}$. Since $\mathcal{D}_{Next_H}$ is a Hall set, $|H| = |\mathcal{D}_{Next_H}|$, so there will be only one element in $O = \mathcal{D}_{Next_H} \setminus H$; let v_e be that element.*

Then, there is no Hamiltonian Circuit of the graph $G = (V, E)$ such that the successor of v_s is v_e.

Proof. Let $J = H \cap \mathcal{D}_{Next_H}$; by the assumptions in the theorem, $H = J \cup \{h_s\}$ and $\mathcal{D}_{Next_H} = J \cup \{v_e\}$.

By definition of $\mathcal{D}_{Next_H}$, the successor of each node in H is one element in $\mathcal{D}_H = J \cup \{v_e\}$. Since no two nodes can have the same successor, the successor of h_s cannot be v_e, otherwise the successors of all the elements in J would be other elements in J, making the set J an isolated subset (see Figure 1). □

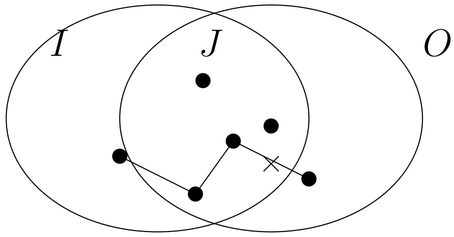

Figure 1: Division of nodes into the three sets as in Theorem 4.2, and possible pruning that can be obtained. Picture drawn with ASPECT [19].

Theorem 4.3. *Let $H = \{h_1, \dots, h_k\}$ be set of nodes with cardinality $k < n$, and let $\mathcal{D}_{Next_H}$ be a Hall set.*

Let $J = H \cap \mathcal{D}_{Next_H}$, $I = H \setminus \mathcal{D}_{Next_H}$ and $O = \mathcal{D}_{Next_H} \setminus H$.

Then, in any Hamiltonian Circuit:

1. *The successor of an element in I is either an element in J or in O.*
2. *The successor of an element in J is either en element in J or in O.*
3. *The successor of an element in $V \setminus H$ is in $V \setminus H$ or in I.*

Proof. Note that since $\mathcal{D}_{Next_H}$ is a Hall set, the successor of each element in $V \setminus H$ cannot be in the set $\mathcal{D}_{Next_H}$ (this is exactly the propagation performed by the `alldifferent` constraint); this proves item 3.

Items 1 and 2 follow immediately from the definition of $\mathcal{D}_{Next_H}$. □

A possible propagation algorithm of the `hamiltonian` constraint could be summarised as follows:

$\mathcal{F} \leftarrow$ `alldifferent` ▷ execute the `alldifferent` propagator; such propagator also returns a family $\mathcal{F}$ of Hall sets.
if $|\mathcal{F}| > 1$ **then**
  for all $H \in \mathcal{F}$ **do**
    $I \leftarrow H \setminus \mathcal{D}_{Next_H}$
    if $|I| = 1$ **then**
      $O \leftarrow \mathcal{D}_{Next_H} \setminus H$
      let $I = \{i\}$
      let $O = \{o\}$
      $Next_i \neq o$
    end if
  end for
end if

The complexity of this propagator is dominated by the invocation of the `alldifferent` constraint, that amounts at $O(n^{2.5})$ or $O(n \log n)$ depending on the achieved level of consistency. The set differences $H \setminus \mathcal{D}_H$ and $\mathcal{D}_H \setminus H$ can be computed in linear time, if the sets H and $\mathcal{D}_H$ are sorted, so the complexity of the `hamiltonian` constraint does not increase with respect to the `alldifferent` constraint.

A second level of pruning can be obtained in the same situation (i.e., when $|H \setminus \mathcal{D}_{Next_H}| = 1$ and H is a Hall set) by considering that from the initial node one cannot get to the final node of a Hall set unless all the nodes in the set are visited. The propagator can be implemented in a similar way to the `circuit` constraint: when a partial path inside the Hall set H has been assigned (i.e., a sequence of variables have become ground and they represent a partial path), reaching the final node is forbidden unless all the nodes in H have been visited. The following pseudo-code depicts the algorithm performed by the propagator; it is invoked with $circuit\_in\_Hall\_set\_prop(i, o, 0, H, Next)$, where i is the initial node in the Hall set (as in the previous algorithm), o is the output node.

function CIRCUIT_IN_HALL_SET_PROP($Start, End, Len, H, Next$)
  if $Next_{Start}$ is ground **then**
    circuit_in_Hall_set_prop($Next_{Start}, End, Len + 1, H, Next$)
  else if $Len < |H|$ **then**
    remove End from $Dom(Next_{Start})$
    suspend waiting for $Next_{Start}$ to become ground
  else
    return $true$
  end if
end function

5. Experimental Evaluation

In this section, we present the experiments conducted on the integration of the `alldifferent` and `circuit` constraints, as previously introduced.

The experiments focus on evaluating the performance of three different constraint models: `alldiff_circuit`, and two variants of the newly proposed constraint, namely `hcc_nopath`, and `hcc_path`, for solving the Hamiltonian Cycle Problem. All algorithms are implemented in the ECL^{i}PS^e CLP language [11].

The three constraint models differ in their approach as follows:

- `alldiff_circuit`: this model uses the `alldifferent` and `circuit` constraints from the ECL^{i}PS^e libraries. Specifically, the `alldifferent` implementation is the one inspired by the algorithm proposed by Régin [2] and is provided by the `ic_global` library, while the implementation of the `circuit` constraint is from the `ic` library;
- `hcc_nopath`: this variant also applies the `circuit` constraint from the `ic` library but it implements, in the `alldifferent` constraint from the `ic_global` library, the pruning strategy introduced in the previous section;
- `hcc_path`: this model builds on `hcc_nopath` by adding path-based pruning within the `alldifferent` constraint, specifically within Hall sets, as detailed in the $circuit\_in\_Hall\_set\_prop$ function.

All tests were run on ECL^{i}PS^e v. 7.1beta, build #13, on AMD EPYC 9454 running at 2.75GHz, using only one core and with 4GB of reserved memory. The ECL^{i}PS^e Constraint Programming System is distributed as open-source software[1].

We evaluated the effectiveness of the proposed constraints on two types of graph: *uniform* and *clustered*. In uniform graphs, a fixed number of nodes, denoted as N, and a connection probability p are specified. For each pair of distinct nodes, a directed edge is established between them with probability p, excluding self-loops, meaning no edge connects a node to itself. Clustered graphs, on the other hand, are characterized by the number of clusters C. Nodes are equally distributed among the clusters, with each cluster consisting of N/C nodes. Within each cluster, nodes are interconnected by directed edges with probability p. Additionally, each cluster is connected to two other clusters by exactly four directed edges, two edges for each cluster, ensuring a cyclic structure.

The values used for N ranged from 100 to 1000, in increments of 100, while for p, ranged from 0.05 to 0.95 in increments of 0.05. For each combination of these parameters, 20 random graphs were generated, for a total of 3,800 uniform graphs. For the clustered graphs, we tested four different values of C: 5, 10, 20 and 30 that is added to the combinations of N and p of uniform graphs. In this way, the total number of clustered graphs tested was 15,200.

Results for uniform graph are shown in Figure 2. The x-axis represents the probability threshold p, which affects the graph's density. The y-axis is divided into two metrics: the difference in the number of solved instances relative to the reference algorithm `alldiff_circuit` (shown with bars - higher

[1] https://eclipseclp.org/

is better), and the CPU time required for execution measured in seconds (shown with lines - lower is better). Each point on the lines represents the geometric mean of the solving time of the instances generated with that specific probability threshold.

The reference algorithm, `alldiff_circuit`, consistently performs with slightly lower execution times compared to the two new variants `hcc_nopath` and `hcc_path`, across most of the probability thresholds. The number of successfully solved instances also does not differ drastically among the algorithms, though `alldiff_circuit` leads by a small margin. This performance difference may be attributed to the absence of conditions under which the constraint propagation of `hcc_nopath` and `hcc_path` can be effectively performed. This could be due to the structure of the uniform graphs themselves, or the variable selection (the variable with the smallest domain size is selected first) and value selection (values are tried in increasing order) strategies employed, which may not favor the propagation conditions necessary for optimization.

This hypothesis is supported by the results obtained on the clustered graphs, presented in Figure 3. The analysis was conducted in a manner consistent with that used for uniform graphs to ensure comparability. As the probability threshold increases, both our variants achieve similar and significant reductions in solving time, ranging from 20% to 28%, compared to the reference algorithm `alldiff_circuit`. This decrease in solving time is accompanied by a higher number of solved instances, with a 5% increase when $p = 0.95$.

Among our two variants, no significant differences are observed in their solving times as both exhibit comparable improvements with respect to `alldiff_circuit`. However, while both demonstrate better performance, the `hcc_nopath` variant shows a slightly smaller increase in the number of solved instances.

Finally, we examined the behavior of our algorithms as the number of clusters in the graph varied. The results, shown in Figure 4, focus on instances with a probability $p > 0.5$, given their relevance assessed from previous analyses. The data shows a clear trend where fewer clusters result in a 26% reduction in average solving time. However, as the number of clusters increases, approximating the structure of a uniform graph, the effectiveness of our constraints decreases.

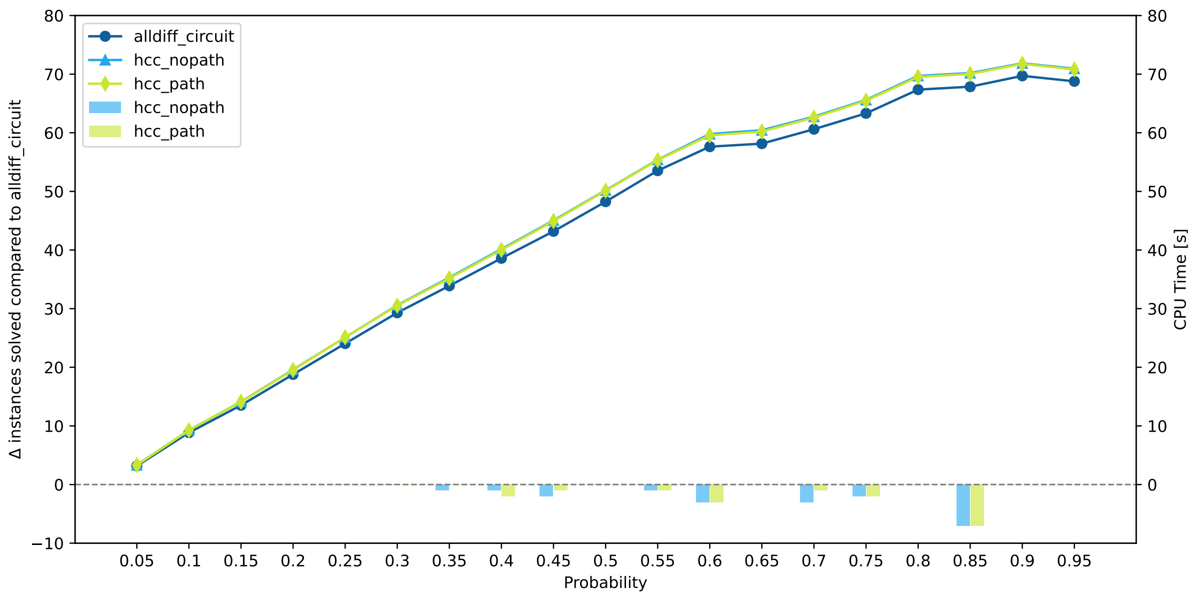

Figure 2: Comparison of the performance of the reference algorithm `alldiff_circuit` and the two new variants (`hcc_nopath`, and `hcc_path`) on **uniform** graphs, varying the connection **probability** p. The y-axis includes two metrics: the number of additional solved instances compared to `alldiff_circuit` (bars, higher is better) and the CPU time in seconds (lines, lower is better).

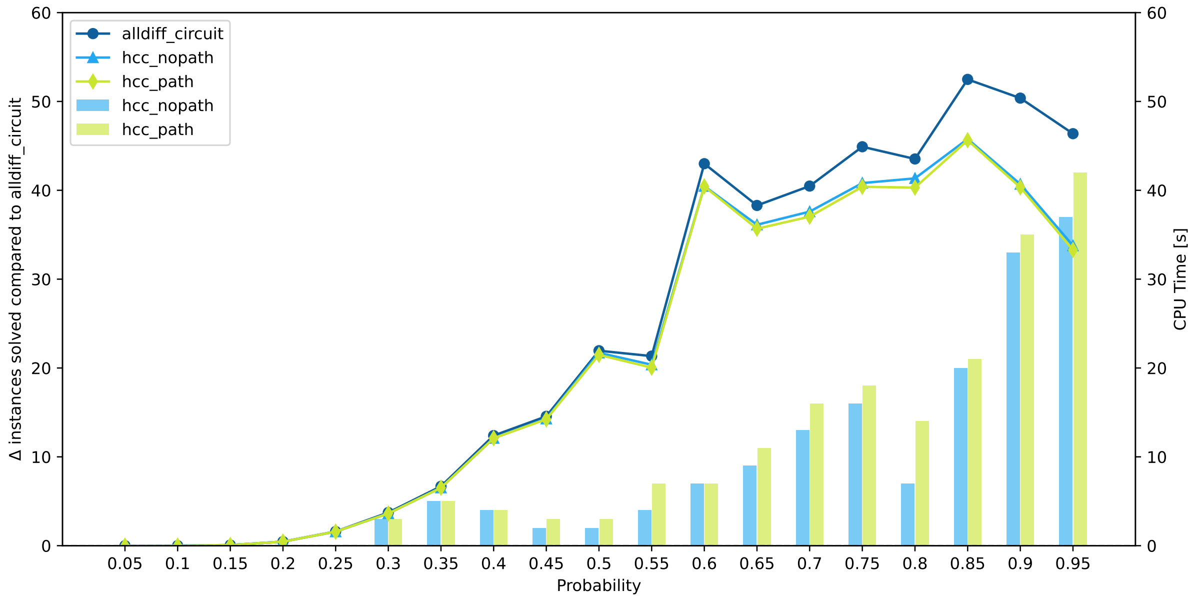

Figure 3: Comparison of the performance of the reference algorithm `alldiff_circuit` and the two new variants (`hcc_nopath`, and `hcc_path`) on **clustered** graphs, varying the connection **probability** p. The y-axis includes two metrics: the number of additional solved instances compared to `alldiff_circuit` (bars, higher is better) and the CPU time in seconds (lines, lower is better).

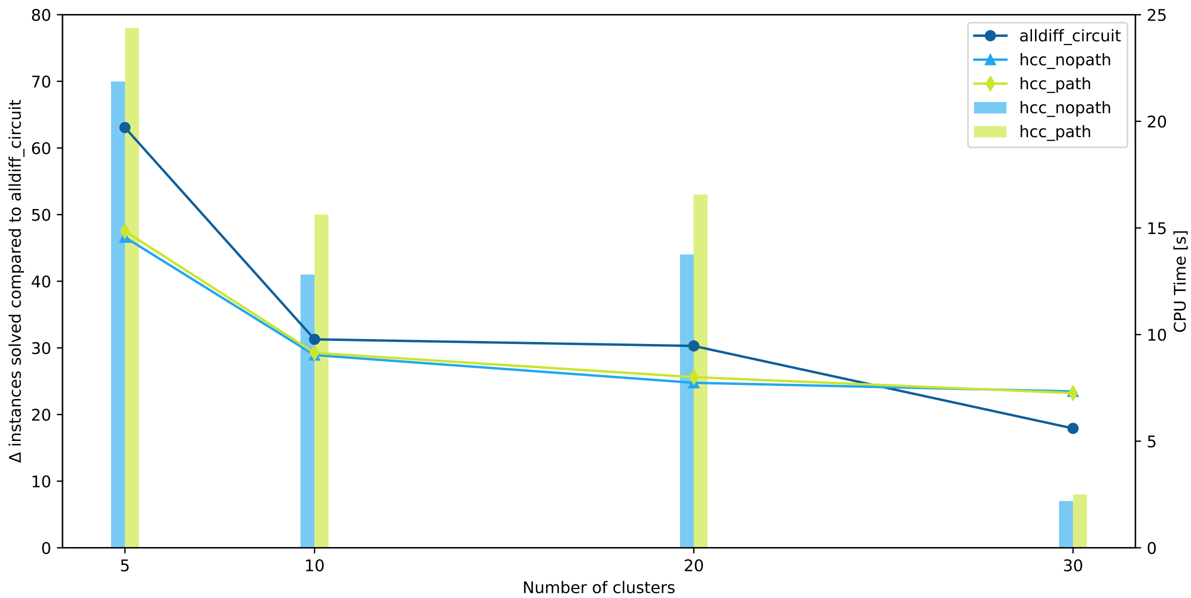

Figure 4: Comparison of the performance of the reference algorithm `alldiff_circuit` and the two new variants (`hcc_nopath`, and `hcc_path`) on **clustered** graphs, varying the **number of clusters** C. The y-axis includes two metrics: the number of additional solved instances compared to `alldiff_circuit` (bars, higher is better) and the CPU time in seconds (lines, lower is better).

6. Conclusions

In this paper, we proposed a combination of the famous `alldifferent` and `circuit` constraints that reuses the data structures used to propagate the `alldifferent` in order to obtain further pruning for the `circuit` constraint. The negligible overhead makes it a viable combination to improve the propagation.

Experimental results on the Hamiltonian Circuit Problem show that several instances are more efficiently solved by the combination than by the two separate constraints; the speedup is more significant in instances having a clustered structure.

In future work, we plan to investigate other types of integrations in routing problems; particularly promising could be the integration with constraints tailored to improve the pruning of Euclidean Travelling Salesperson problems [20, 21]. We also plan to extend the experimentation, compare with other implementations of the two constraints considered in this research, and study how the search strategy influences the effectiveness of the solution process.

Acknowledgments

Research funded by the Italian Ministry of University and Research through PNRR - M4C2 - Investimento 1.3 (Decreto Direttoriale MUR n. 341 del 15/03/2022), Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 8 "Pervasive AI", funded by the European Union under the NextGeneration EU programme". Alessandro Bertagnon and Marco Gavanelli are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

References

[1] D. Aineto, R. De Benedictis, M. Maratea, M. Mittelmann, G. Monaco, E. Scala, L. Serafini, I. Serina, F. Spegni, E. Tosello, A. Umbrico, M. Vallati (Eds.), Proceedings of the International Workshop on Artificial Intelligence for Climate Change, the Italian workshop on Planning and Scheduling, the RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and the Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (AI4CC-IPS-RCRA-SPIRIT 2024), co-located with 23rd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[2] J. Régin, A filtering algorithm for constraints of difference in CSPs, in: B. Hayes-Roth, R. E. Korf (Eds.), Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1., AAAI Press / The MIT Press, 1994, pp. 362–367.

[3] J. Puget, A fast algorithm for the bound consistency of alldiff constraints, in: J. Mostow, C. Rich (Eds.), Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA, AAAI Press / The MIT Press, 1998, pp. 359–366. URL: http://www.aaai.org/Library/AAAI/1998/aaai98-051.php.

[4] K. Mehlhorn, S. Thiel, Faster algorithms for bound-consistency of the sortedness and the alldifferent constraint, in: R. Dechter (Ed.), Principles and Practice of Constraint Programming - CP 2000, 6th International Conference, Singapore, September 18-21, 2000, Proceedings, volume 1894 of *Lecture Notes in Computer Science*, Springer, 2000, pp. 306–319. URL: https://doi.org/10.1007/3-540-45349-0_23. doi:10.1007/3-540-45349-0_23.

[5] A. López-Ortiz, C. Quimper, J. Tromp, P. van Beek, A fast and simple algorithm for bounds consistency of the alldifferent constraint, in: G. Gottlob, T. Walsh (Eds.), IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003, Morgan Kaufmann, 2003, pp. 245–250. URL: http://ijcai.org/Proceedings/03/Papers/036.pdf.

[6] X. Zhang, Q. Li, W. Zhang, A fast algorithm for generalized arc consistency of the alldifferent constraint, in: J. Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, ijcai.org, 2018, pp. 1398–1403. URL: https://doi.org/10.24963/ijcai.2018/194. doi:10.24963/ijcai.2018/194.

[7] W.-J. Van Hoeve, The alldifferent constraint: A survey, in: Sixth Annual Workshop of the ERCIM Working Group on Constraints, 2001. Http://www.arxiv.org/html/cs/0110012.

[8] M. Leconte, A bounds-based reduction scheme for difference constraints, in: Constraint-96, Second International Workshop on Constraint-based Reasoning, Key West, Florida, 1996.
[9] P. Hall, On representatives of subsets, Journal of the London Mathematical Society 10 (1935) 26–30.
[10] N. Beldiceanu, E. Contejean, Introducing global constraints in CHIP, Math. Comput. Model. 20 (1994) 97–123.
[11] J. Schimpf, K. Shen, Ecl^{i}ps^{e} - from LP to CLP, TPLP 12 (2012) 127–156.
[12] Y. Caseau, F. Laburthe, Solving small TSPs with constraints, in: L. Naish (Ed.), Logic Programming, Proceedings of the Fourteenth International Conference on Logic Programming, Leuven, Belgium, July 8-11, 1997, MIT Press, 1997, pp. 316–330.
[13] L. G. Kaya, J. N. Hooker, A filter for the circuit constraint, in: F. Benhamou (Ed.), Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006, Proceedings, volume 4204 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 706–710.
[14] K. G. Francis, P. J. Stuckey, Explaining circuit propagation, Constraints 19 (2014) 1–29. URL: https://doi.org/10.1007/s10601-013-9148-0. doi:10.1007/s10601-013-9148-0.
[15] N. Isoart, J. Régin, Integration of structural constraints into TSP models, in: T. Schiex, S. de Givry (Eds.), Principles and Practice of Constraint Programming - 25th International Conference, CP 2019, Stamford, CT, USA, September 30 - October 4, 2019, Proceedings, volume 11802 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 284–299. URL: https://doi.org/10.1007/978-3-030-30048-7_17. doi:10.1007/978-3-030-30048-7_17.
[16] N. Isoart, J. Régin, A linear time algorithm for the k-cutset constraint, in: L. D. Michel (Ed.), 27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021, volume 210 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 29:1–29:16. URL: https://doi.org/10.4230/LIPIcs.CP.2021.29. doi:10.4230/LIPICS.CP.2021.29.
[17] R. M. Karp, Reducibility among Combinatorial Problems, Springer US, Boston, MA, 1972, pp. 85–103. URL: https://doi.org/10.1007/978-1-4684-2001-2_9. doi:10.1007/978-1-4684-2001-2_9.
[18] C. Bessiere, E. Hebrard, B. Hnich, T. Walsh, The complexity of global constraints, in: D. L. McGuinness, G. Ferguson (Eds.), Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA, AAAI Press / The MIT Press, 2004, pp. 112–117. URL: http://www.aaai.org/Library/AAAI/2004/aaai04-018.php.
[19] A. Bertagnon, M. Gavanelli, F. Zanotti, ASPECT: Answer Set rePresentation as vEctor graphiCs in laTex, in: A. Dovier, A. Formisano (Eds.), Proceedings of the 38th Italian Conference on Computational Logic, Udine, Italy, June 21-23, 2023, volume 3428 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3428/paper3.pdf.
[20] A. Bertagnon, M. Gavanelli, Improved filtering for the euclidean traveling salesperson problem in CLP(FD), in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 1412–1419. doi:10.1609/AAAI.V34I02.5498.
[21] E. Bellodi, A. Bertagnon, M. Gavanelli, R. Zese, Improving the efficiency of euclidean TSP solving in constraint programming by predicting effective nocrossing constraints, in: M. Baldoni, S. Bandini (Eds.), AIxIA 2020 - Advances in Artificial Intelligence - XIXth International Conference of the Italian Association for Artificial Intelligence, Virtual Event, November 25-27, 2020, Revised Selected Papers, volume 12414 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 318–334. doi:10.1007/978-3-030-77091-4_20.