Chapter 7
Linear Programming Based Algorithms

7.1 Introduction

In Chapter 4, we showed that when a graph is embedded in a suitably constructed Markov decision process, the associated convex domain of discounted occupational measures is a polyhedron with extreme points corresponding to all spanning subgraphs of the given graph. Furthermore, from Theorem 4.1 we learned that a simple cut of the above domain yields a polyhedron the extreme points of which correspond to only two possible types: Hamiltonian cycles and convex combinations of short and noose cycles. These properties, naturally, suggest certain algorithmic approaches to searching for Hamiltonian cycles.

In this chapter, we present two methods to solve the HCP: the *branch and fix method*, and the *Wedged-MIP heuristic*. Both methods take advantage of the Markov decision process embedding outlined in Chapter 4. The branch and fix method is implemented in MATLAB and results that are supplied demonstrate the potential of this model. The Wedged-MIP heuristic is implemented in IBM ILOG OPL-CPLEX and succeeds in solving large graphs, including two of the large test problems given on the TSPLIB website maintained by the University of Heidelberg [98]. Both of these methods operate in the space of discounted occupational measures, but similar methods could be developed for the space of limiting average occupational measures.

Here, we continue exploiting properties of the space of discounted occupational measures in the Markov decision process associated with a graph G, as outlined in Chapter 4. In particular, we apply the non-standard branch and bound method of Filar and Lasserre [50] to Feinberg's embedding of the HCP in a discounted Markov decision process [44] (rather than the limiting average Markov decision process used previously). This embedding has the benefit that the discount parameter does not destroy the sparsity of coefficient matrices to nearly the same extent as did the perturbation parameter ε, used in [50] to replace the underlying probability transitions $p(j|i,a)$ of the MDP by the linearly perturbed transitions $p^{\varepsilon}(j|i,a)$. We refer to the method that arises from this embedding as the branch and fix method[1].

We show that in the present application, the appropriate sub-space of discounted occupational measures is synonymous with a polytope $\widetilde{\mathcal{X}}_\beta$ defined by only $N+1$ equality constraints and nonnegativity constraints. Using results in Theorems 4.1 and 4.2 about the structure of extreme points of $\widetilde{\mathcal{X}}_\beta$, we conjecture that Hamiltonian cycles will be found far earlier, and the resulting *logical branch and fix tree* will have fewer branches than that for more common polytopes. The logical branch and fix (B&F) tree that arises from the branch and fix method is a rooted tree. The root of the logical B&F tree corresponds to the original graph G, and each branch corresponds to a certain fixing of arcs in G. Then, a branch forms a pathway from the root of the logical B&F tree to a leaf. These leaves correspond to particular subdigraphs of G, which may or may not contain Hamiltonian cycles. At the maximum depth of the logical B&F tree, each leaf corresponds to a subdigraph for which there is exactly one arc emanating from every vertex. We refer to subdigraphs of this type as *spanning 1-out-regular subdigraphs* of G. Leaves at a shallower level correspond to subdigraphs in which there are multiple arcs emanating from at least one vertex.

The set of all spanning 1-out-regular subdigraphs has a one-to-one correspondence with the set of all deterministic policies in G. Even for graphs with bounded out-degree, this represents a set with non-polynomial cardinality. Cubic graphs, for example, have 3^N distinct deterministic policies. Hence, it is desirable to be able to fathom branches early, and consequently restrict the number of leaves in the logical B&F tree. The special structure of the extreme points of $\widetilde{\mathcal{X}}_\beta$ usually enables us to identify a Hamiltonian cycle before obtaining a spanning 1-out-regular subdigraph, limiting the depth of the logical B&F tree. We achieve significant improvements by introducing into the branch and fix method additional feasibility constraints as bounds, and logical checks that allow us to fathom branches early. This further limits the depth of the logical B&F tree.

The resulting method is guaranteed to solve the HCP in finitely many iterations. While the worst case may involve examination of exponentially many branches, empirically we show that the number of branches required to find a Hamiltonian cycle is generally reduced to a tiny fraction of the total number of deterministic policies. For example, a 24-vertex Hamiltonian cubic graph has $3^{24} \approx 3 \times 10^{11}$ possible choices for deterministic policies, but the algo-

[1] Since the speed of convergence depends more on arc fixing features than on bounds, the name *branch and fix* (or B&F) method is more appropriate than *branch and bound.*

rithm finds a Hamiltonian cycle by examining only 28 branches. We observe that Hamiltonian graphs perform better than non-Hamiltonian graphs, as they typically have many Hamiltonian cycles spread throughout the logical B&F tree, and only one needs to be found. However, even in non-Hamiltonian graphs we demonstrate that the algorithm performs well. For instance, a 28-vertex non-Hamiltonian cubic graph has $3^{28} \approx 2 \times 10^{13}$ possible choices for deterministic policies, but the algorithm terminates after investigating only 11,708 branches. This example highlights the ability of the B&F method to fathom branches early, allowing us to ignore, in this case, 99.99999995% of the potential branches.

In addition to the basic branch and fix method, we develop and compare several branching methods for traversing the logical B&F tree that may find Hamiltonian cycles quicker in certain graphs, and propose additional constraints that can find infeasibility at an earlier depth in the logical B&F tree. We provide experimental results demonstrating the significant improvement achieved by these additions. We also demonstrate that $\widetilde{\mathcal{X}}_\beta$ can be a useful polytope in many other optimisation algorithms. In particular we use $\widetilde{\mathcal{X}}_\beta$, along with the additional constraints, in a mixed integer programming model that can solve extremely large graphs using commercially available software such as CPLEX. Finally, we present solutions of four large non-regular graphs, with 250, 500, 1000 and 2000 vertices respectively, which are obtained by this model.

7.2 Branch and Fix Method

In this section, we describe the branch and fix method and some of the techniques used within that branch and fix method that are designed to help limit the size of the logical branching tree. The original source for this algorithm is Ejov *et al.* [31] and the presentation here is based on the PhD thesis of Haythorpe [62].

Outline of The Branch and Fix Method

In view of the fact that it is only 1-randomised policies that prevent standard simplex methods from finding a Hamiltonian cycle, it has been recognised for some time that branch and bound-type methods can be used to eliminate the possibility of arriving at these undesirable extreme points (see, for instance, Filar and Lasserre [50]). However, the method reported in Filar and Lasserre [50] uses an embedding in a long-run average MDP, with a perturbation of transition probabilities that introduces a small parameter in most of the $p(j|i,a)$ coefficients of variables in linear constraints (4.18), thereby leading to loss of sparsity. Furthermore, the method in Filar and Lasserre [50] was never implemented fully, or tested beyond a few simple examples.

Theorem 4.1 indicates that 1-randomised policies induced by extreme points of $\widetilde{\mathcal{X}}_\beta$ are less prevalent than might have been conjectured, since they cannot be constructed from convex combinations of just any two deterministic policies. This provides motivation for testing algorithmic approaches based on successive elimination of arcs that could be used to construct these convex combinations. Since our goal is to find an extreme point $\boldsymbol{x}_e \in \widetilde{\mathcal{X}}_\beta$ such that

$$\boldsymbol{f} = M^{-1}(\boldsymbol{x}_e) \in \mathcal{F}_{\mathcal{D}},$$

we have a number of degrees of freedom in designing an algorithm. In particular, different linear objective functions can be chosen at each stage of the algorithm, the parameter $\beta \in (0,1)$ can be adjusted, and $\mu \in (0, 1/N)$ can be chosen small but not so small as to cause numerical difficulties. The latter parameter needs to be positive to ensure that the inverse map M^{-1} is well-defined. In the experiments reported here, we choose μ to be $1/N^2$.

The branch and fix method is as follows. We solve a sequence of linear programs—two at each branching point of the logical B&F tree—with the generic structure

$$\begin{aligned} &\min L(\boldsymbol{x}) \\ &\text{subject to} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (7.1) \\ &\boldsymbol{x} \in \widetilde{\mathcal{X}}_\beta, \text{and additional constraints, if any, on arcs fixed earlier.} \end{aligned}$$

Step 1—Initiation. We solve the original LP (7.1) without any additional constraints and with some choice of an objective function $L(\boldsymbol{x})$, to obtain an optimal basic feasible solution $\boldsymbol{x}_0$. We then find $\boldsymbol{f}_0 = M^{-1}(\boldsymbol{x}_0)$. If $\boldsymbol{f}_0 \in \mathcal{F}_{\mathcal{D}}$, we stop, the policy $\boldsymbol{f}_0$ identifies a Hamiltonian cycle. Otherwise, $\boldsymbol{f}_0$ is a 1-randomised policy.

Step 2—Branching. We use the 1-randomised policy $\boldsymbol{f}_0$ to identify the splitting vertex i, and two arcs (i, j_1) and (i, j_2) corresponding to the single randomisation in $\boldsymbol{f}_0$. If there are d arcs $\{(i, a_1), \ldots, (i, a_d)\}$ emanating from vertex i, we construct d subdigraphs: $G_1, G_2, \ldots, G_d$, where in G_k the arc (i, a_k) is the only arc emanating from vertex i. These graphs are identical to the original graph G at all other vertices. In this process we, by default, fix an arc in each G_k.

Step 3—Fixing. In many subdigraphs, the fixing of one arc implies that other arcs may also be fixed[2], without a possibility of unintentionally elim-

[2] This frequently happens in the case of cubic graphs that supplied many of our test examples. For instance, see Figure 7.2.

inating a Hamiltonian cycle containing already fixed arcs that are part of a Hamiltonian cycle in the current subdigraph. Later in this section, we describe four checks for determining additional arcs that can be fixed. Once we identify these arcs, we also fix them at this step.

Step 4—Iteration. We solve a second LP (with the objective function (7.9)) to determine if (4.19) is still satisfied with the current fixing of arcs. If so, we repeat Step 1 with the LP (7.1) constructed for the graph at the current branching point of the logical B&F tree, with additional constraints derived in (7.5) and (7.6) below. This branching point may correspond to $G_1, G_2, \ldots, G_d$, or to a sub-graph constructed from one of these with the help of additional arc fixing[3].

We now briefly discuss the construction of additional constraints alluded to in Step 4 of the B&F. If $\boldsymbol{f}$ is a Hamiltonian policy, $\boldsymbol{x} = M(\boldsymbol{f})$, and $\mu = 0$, then we can easily check that $\boldsymbol{x}$ satisfies (4.18)–(4.20) and, for $k = 0, \ldots, N-1$,

$$x_{i_k i_{k+1}} = \sum_{a \in \mathcal{A}(i_k)} x_{i_k a} = \frac{\beta^k}{1-\beta^N} \tag{7.2}$$

where (i_k, i_{k+1}) is the kth arc on the Hamiltonian cycle traced out by $\boldsymbol{f}$. This immediately suggests lower and upper bounds on sums of the $\boldsymbol{x}$ variables corresponding to arcs emanating from the heads of fixed arcs. This is because if $i_{k+1} \neq 1$,

$$\sum_{a \in \mathcal{A}(i_{k+1})} x_{i_{k+1} a} - \beta x_{i_k i_{k+1}} = 0. \tag{7.3}$$

If $i_{k+1} = 1$, then $k+1 = N$ and we have

$$-\beta^N \sum_{a \in \mathcal{A}(1)} x_{1a} + \beta x_{i_{N-1},1} = 0. \tag{7.4}$$

For $\mu > 0$, analogous, but more complex, expressions for the preceding sums can be derived and the relationship (7.3) between these sums at successive vertices on the Hamiltonian cycle, for $i_{k+1} \neq 1$, is simply

$$\sum_{a \in \mathcal{A}(i_{k+1})} x_{i_{k+1} a} - \beta x_{i_k i_{k+1}} = \mu. \tag{7.5}$$

If the fixed arc is the final arc $(i_N, 1)$, we have

[3] As is typical with branching methods, decisions guiding which branch to select first are important and open to alternative heuristics. We investigate five possible branching methods later in this section.

$$-\beta^N \sum_{a\in\mathcal{A}(1)} x_{1a} + \beta x_{i_{N-1},1} = \frac{\mu\beta(1-\beta^{N-1})}{1-\beta}. \tag{7.6}$$

We derive equation (7.5) by simply inspecting the form of (4.18). For (7.6), we know from (4.18) that

$$\sum_{a\in\mathcal{A}(1)} x_{1a} - \beta x_{i_{N-1},1} = 1-(N-1)\mu,$$

and therefore

$$\beta x_{i_{N-1},1} = \sum_{a\in\mathcal{A}(1)} x_{1a} - 1 + (N-1)\mu. \tag{7.7}$$

Then, we substitute (7.7) into the left-hand side of (7.6) to obtain

$$-\beta^N \sum_{a\in\mathcal{A}(1)} x_{1a} + \beta x_{i_{N-1},1} = (1-\beta^N) \sum_{a\in\mathcal{A}(1)} x_{1a} - 1 + (N-1)\mu. \tag{7.8}$$

Finally, we substitute (4.19) into (7.8) to obtain

$$\begin{aligned} &-\beta^N \sum_{a\in\mathcal{A}(1)} x_{1a} + \beta x_{i_{N-1},1} \\ &= (1-\beta^N)\frac{(1-(N-1)\mu)(1-\beta)+\mu(\beta-\beta^N)}{(1-\beta)(1-\beta^N)} - 1 + (N-1)\mu \\ &= \frac{\mu\beta(1-\beta^{N-1})}{1-\beta}, \end{aligned}$$

which coincides with (7.6).

Structure of the underlying LP in the branch and fix method

At the initiation step of the B&F method, we solve a feasibility problem of satisfying constraints (4.18)–(4.20). This allows us to determine on which vertex to begin branching.

At every branching point of the logical B&F tree other than the root, we solve an additional LP that attempts to determine if we need to continue exploring the current branch. As the algorithm evolves along successive branching points of the logical B&F tree, we have additional information about which arcs have been fixed. This permits us to perform tests to check the possibility of finding a Hamiltonian cycle containing these fixed arcs. If we determine that it is impossible, we fathom that branching point of the logical B&F tree and no further exploration of that branch is required. For instance, suppose that all fixed arcs belong to a set $\mathcal{U}$. Let the objective function of a second LP[4] be

$$L(\boldsymbol{x}) = \sum_{a\in\mathcal{A}(1)} x_{1a}, \tag{7.9}$$

and minimise (7.9) subject to constraints (4.18) and (4.20) together with equations (7.5) and (7.6) providing additional constraints for each arc in $\mathcal{U}$. If the minimum $L^*(\boldsymbol{x})$ fails to reach the level defined by the right-hand side of the now omitted constraint (4.19) of $\widetilde{\mathcal{X}}_\beta$, or if the constraints are infeasible, then there exists no Hamiltonian cycle that uses all the arcs of $\mathcal{U}$, and we fathom the current branching point of the logical B&F tree. Otherwise, we solve the LP (7.1) with the objective function[5]

$$L(\boldsymbol{x}) = \sum_{(i,j)\in\mathcal{U}} \{ \sum_{a\in\mathcal{A}(j)} x_{ja} - \beta \sum_{a\in\mathcal{A}(i)} x_{ia} \}, \tag{7.10}$$

and with no additional constraints beyond those in $\widetilde{\mathcal{X}}_\beta$. This LP will either find a Hamiltonian cycle, or it will lead to an extreme point $\boldsymbol{x}'_e$ such that $\boldsymbol{f}' = M^{-1}(\boldsymbol{x}'_e)$ is a new 1-randomised policy. Of course, alternative objective functions $L(\boldsymbol{x})$ could also be considered.

Arc fixing checks

There are a number of logical checks that enable us to fix additional arcs once a decision is taken to fix a particular arc. This is best illustrated with the help of an example. These checks are in the spirit of well-known rules for constructing Hamiltonian cycles (see Tucker [99, Section 8.2]).

Consider the envelope graph (Figure 7.1). Figure 7.2 shows the kind of logical additional arc fixing that can arise.

Check 1: Consider the top-left graph in Figure 7.2. The fixed arcs are $(1,2)$ and $(6,3)$. Since the only arcs that can go to vertex 5 are $(1,5)$, $(4,5)$ and $(6,5)$, we may also fix arc $(4,5)$ as vertices 1 and 6 already have fixed arcs going elsewhere. In this case, we say that arc $(4,5)$ is *free*, whereas arcs $(1,5)$ and $(6,5)$ are not free. In general, if only one free arc enters a vertex, it must be fixed.

[4] Although we call (7.9) the second LP, it is the first LP solved in all iterations other than the initial iteration. Since it is not solved first in the initial iteration, we refer to (7.9) as the second LP.

[5] For simplicity, we are assuming here that $\mathcal{U}$ does not contain any arc going into vertex 1. If such an arc were in $\mathcal{U}$, the objective function (7.10) would have one term consistent with the left-hand side of equation (7.6).

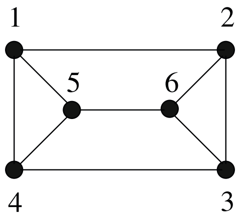

Fig. 7.1: The envelope graph

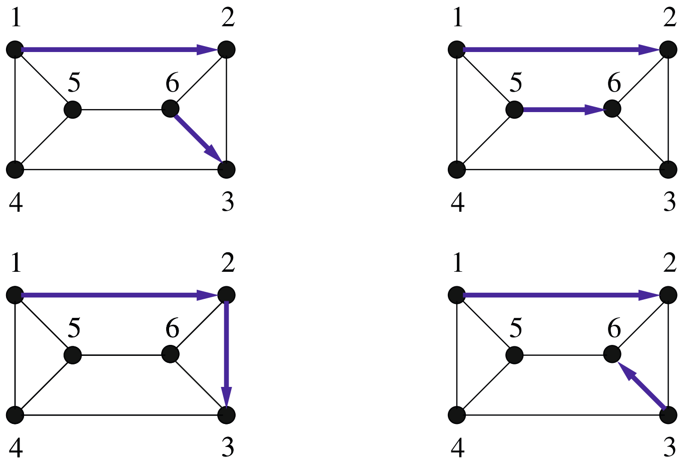

Fig. 7.2: Various arc fixing situations

Check 2: Consider the top-right graph in Figure 7.2. The fixed arcs are $(1, 2)$ and $(5, 6)$. The arcs going to vertex 5 are $(1, 5)$, $(4, 5)$ and $(6, 5)$. We cannot choose $(6, 5)$ as this would create a subcycle of length 2, and vertex 1 already has a fixed arc going elsewhere, so we must fix arc $(4, 5)$. In general, if there are only two free arcs and fixing one arc would create a subcycle, we must fix the other one.

Check 3: Consider the bottom-left graph in Figure 7.2. The fixed arcs are $(1, 2)$ and $(2, 3)$. Since the only arcs that can come from vertex 6 are $(6, 2)$, $(6, 3)$ and $(6, 5)$, we must fix arc $(6, 5)$ as vertices 2 and 3 already have arcs going into them. In this case, we say that arcs $(6, 3)$ and $(6, 5)$ are *blocked*, whereas arc $(6, 2)$ is *unblocked*. In general, if there is only one unblocked arc emanating from a vertex, the arc must be fixed.

Check 4: Consider the bottom-right graph in Figure 7.2. The fixed arcs are $(1, 2)$ and $(3, 6)$. The only arcs that can come from vertex 6 are $(6, 2)$, $(6, 3)$ and $(6, 5)$. We cannot choose $(6, 2)$ because vertex 2 already has an incoming arc, and we cannot choose $(6, 3)$ as this would create a subcycle, so we must fix arc $(6, 5)$. In general, if there are two unblocked arcs emanating from a vertex and fixing one arc would create a subcycle, we must fix the other one.

The branch and bound method given in Filar and Lasserre [50] always finds a Hamiltonian cycle if one exists. While the branch and fix method presented here is in the same spirit as the method in [50], we include a finite convergence proof for the sake of completeness.

Theorem 7.1. *The branch and fix method converges in finitely many steps. In particular, if G is Hamiltonian, the algorithm finds a Hamiltonian cycle; otherwise, the algorithm terminates after fathoming all constructed branches of the logical B&F tree.*

Proof. Given a graph G, at each stage of the algorithm, a splitting vertex is identified and branches are created for all arcs emanating from that vertex. As we consider every arc for this vertex and therefore explore every possibility from this vertex, the branching process cannot prevent the discovery of a Hamiltonian cycle if one exists in the graph G. It then suffices to confirm that none of the checking, bounding, or fixing steps in the branch and fix method can eliminate the possibility of finding a Hamiltonian cycle.

Recall that constraints (4.19), (7.5) and (7.6) are shown to be satisfied by all Hamiltonian cycles. Then, for a particular branching point, if the minimum value of (7.9) constrained by (4.18), (4.20), (7.5) and (7.6) cannot achieve the value given in (4.19), or if the constraints are infeasible, there cannot be any Hamiltonian cycles remaining in the subdigraph. Therefore, fathoming the branch due to the second LP (with the objective function (7.9)) cannot eliminate any Hamiltonian cycles. Checks 1–4 above are designed to ensure that at least one arc goes into and comes out of each vertex (while preventing the formation of subcycles), by fixing one or more arcs in situations where any other choice would violate this requirement. Since this is a requirement for all Hamiltonian cycles, it follows that the arc fixing procedures previously described cannot eliminate any Hamiltonian cycles.

The B&F method continues to search the tree until either a Hamiltonian cycle is found, or all constructed branches are fathomed. Since none of the steps in the B&F method can eliminate the possibility of finding a Hamiltonian cycle, we are guaranteed to find one of the Hamiltonian cycles in G. If all branches of the logical B&F tree are fathomed without finding any Hamiltonian cycles, we can conclude that G is non-Hamiltonian. □

While the branch and fix method only finds a single Hamiltonian cycle, it is possible to find all Hamiltonian cycles by simply recording each Hamiltonian cycle when it is found, and then continuing to search the branch and fix tree rather than terminating.

Corollary 7.1. *The depth of the logical B&F tree has an upper bound of N.*

Proof. Since at each branching point of the B&F we branch on all arcs emanating from a vertex, it follows that once an arc (i, j) is fixed, no other arcs emanating from vertex i can be fixed. Then, at each level of the branch and fix tree, a different vertex is branched on. After N levels, all vertices will have exactly one arc fixed, and either a Hamiltonian cycle will be found, or the relevant LP will be infeasible and we will fathom that branch. □

In practice, the arc fixing checks ensure that we never reach this upper bound, as we will certainly fix multiple arcs at branching points corresponding to subdigraphs where only few unfixed arcs remain.

7.3 An Algorithm that Implements the Branch and Fix Method

In Section 7.2, we described the branch and fix method for the HCP and proved its finite convergence. Here, we present a recursive algorithm that implements the method in pseudocode format, with separate component algorithms for the arc fixing checks and for solving the second LP. The input variable *fixed arcs* is initially input as an empty vector, as no arcs are fixed at the commencement of the algorithm. The output term *HC* may either be a Hamiltonian cycle found by the branch and fix method, or a message that no Hamiltonian cycle was found.

```
Input: $G$, fixed arcs
Output: $G$, fixed arcs

begin
        $N \leftarrow$ Size($G$)
        for $i$ from 1 to $N$
            if only one arc $(j, i)$ is free to go into vertex $i$
                  fixed arcs $\leftarrow$ Add arc $(j, i)$ to fixed arcs if it is not already in fixed arcs
            end

            if two arcs $(j, i), (k, i), j \neq k$ are free to go into $i$ and arc $(i, k)$ is in fixed arcs
                  fixed arcs $\leftarrow$ Add arc $(j, i)$ to fixed arcs if it is not already in fixed arcs
            end

            if only one arc $(i, j)$ that emanates from $i$ is unblocked
                  fixed arcs $\leftarrow$ Add arc $(i, j)$ to fixed arcs if it is not already in fixed arcs
            end

            if two arcs $(i, j), (i, k), j \neq k$ that emanate from $i$ are unblocked, and arc $(k, i)$
            is in fixed arcs
                  fixed arcs $\leftarrow$ Add arc $(i, j)$ to fixed arcs if it is not already in fixed arcs
            end
        end
end
```

Algorithm 7.1: Checking algorithm

```
Input: $G, \beta$, fixed arcs
Output: $HC$

begin
    $N \leftarrow$ Size($G$)
    $\mu \leftarrow 1/N^2$
    function value $\leftarrow$ Algorithm 7.3: Second LP algorithm($G, \beta$, fixed arcs)
    if infeasibility is found or function value $> \dfrac{(1-(N-1)\mu)(1-\beta)+\mu(\beta-\beta^N)}{(1-\beta)(1-\beta^N)}$
        return no HC found
    end
    $\widetilde{\mathcal{X}}_\beta \leftarrow$ constraints (4.18)–(4.20)
    for Each arc in fixed arcs
        if Arc goes into vertex 1
            $\widetilde{\mathcal{X}}_\beta \leftarrow$ Add constraint (7.6)
        else
            $\widetilde{\mathcal{X}}_\beta \leftarrow$ Add constraint (7.5)
        end
    end
    $\boldsymbol{x} \leftarrow$ Solve the LP (7.1) with constraints $\widetilde{\mathcal{X}}_\beta$
    if infeasibility is found
        return no HC found
    elseif a HC is found
        return HC
    end
    splitting vertex $\leftarrow$ Identify which vertex has 2 non-zero entries in $\boldsymbol{x}$
    $d \leftarrow$ Number of arcs emanating from splitting vertex
    for $i$ from 1 to $d$
        $G_d \leftarrow G$ with the d-th arc from splitting vertex fixed
        ($G_d$, new fixed arcs) $\leftarrow$ Algorithm 7.1: Checking algorithm($G_d$, fixed arcs)
        HC $\leftarrow$ Algorithm 7.2: Branch and fix algorithm($G_d, \beta$, new fixed arcs)
        if a HC is found
            return HC
        end
    end
    if a HC is found
        return HC
    else
        return no HC found
    end
end
```

Algorithm 7.2: Branch and fix algorithm

```
Input: $G, \beta$, fixed arcs
Output: function value

begin
    $L(\boldsymbol{x}) \leftarrow$ Sum of all arcs $(1, j)$ emanating from vertex 1
    $\widetilde{\mathcal{X}}_\beta \leftarrow$ constraints (4.18) and (4.20)
    for each arc in fixed arcs
        if arc goes into vertex 1
            $\widetilde{\mathcal{X}}_\beta \leftarrow$ Add constraint (7.6)
        else
            $\widetilde{\mathcal{X}}_\beta \leftarrow$ Add constraint (7.5)
        end
    end
    ($\boldsymbol{x}$,function value) $\leftarrow$ Solve the LP $\min L(\boldsymbol{x})$ subject to $\mathcal{X}_\beta$
end
```

Algorithm 7.3: Second LP algorithm

Numerical results

We implement Algorithms 7.1–7.3 in MATLAB (version 7.4.0.336) and use CPLEX (version 11.0.0) to solve all the linear programming sub-problems. The algorithm is tested on a range of small- to medium-sized graphs. The results are encouraging. The number of branches required to solve each of these problems is only a tiny fraction of the number of deterministic policies. It is clear that non-Hamiltonian graphs require more branches to solve than Hamiltonian graphs of the same size. This is because in a Hamiltonian graph, as soon as a Hamiltonian cycle is found, the algorithm terminates. As there is no Hamiltonian cycle in a non-Hamiltonian graph, the algorithm only terminates after fathoming all generated branches of the logical B&F tree.

We provide a sample of results in Tables 7.1 and 7.2, including a comparison between the number of branches examined and the maximum possible number of branches (deterministic policies), and the running time in seconds. The Dodecahedron, Petersen, and Coxeter graphs, and the Knight's Tour problem are well-known in the literature (see Gross and Yellen [58, p. 12] for the first two, Gross and Yellen [58, p. 225] for the third, and Bondy and Murty [14, p. 241] for the last). The 24-vertex graph is a randomly chosen cubic 24-vertex graph.

Table 7.1: Preliminary results for the branch and fix method

Graph	**Branches**	**Upper bound**	**Time(s)**
Dodecahedron: Ham, $N = 20$, arcs = 60	75	3.4868×10^{9}	2.98
Ham, $N = 24$, arcs = 72	28	2.8243×10^{11}	1.02
8×8 Knight's Tour: Ham, $N = 64$, arcs = 336	failed	9.1654×10^{43}	$>$ 12 hrs
Petersen: non-Ham, $N = 10$, arcs = 30	53	5.9049×10^{4}	0.99
Coxeter: non-Ham, $N = 28$, arcs = 84	11589	2.2877×10^{13}	593.40

In Table 7.2, in the first column we refer to the sets of cubic graphs with the prescribed number of vertices in the first column, in the second column we give the average number of branches examined by the branch and fix method, with the average taken over all graphs in the corresponding set of graphs. In the last three columns, we present the minimum and maximum branches examined over the set of graphs, and the average running time taken to solve the graphs in the corresponding class.

More specifically, we consider all 10-vertex cubic graphs, of which there are 17 Hamiltonian and 2 non-Hamiltonian graphs, and all 12-vertex cubic graphs, of which there are 80 Hamiltonian and 5 non-Hamiltonian graphs. We randomly generate 50 cubic graphs of size $N = 20, 30, 40$ and 50. All of the randomly

Table 7.2: Performance of the branch and fix method over cubic graphs

Type of graphs	**Average branches**	**Minimum branches**	**Maximum branches**	**Average time(s)**
Hamiltonian, N = 10	2.1	1	4	0.08
Hamiltonian, N = 12	3.4	1	10	0.14
Non-Hamiltonian, N = 10	32.5	12	53	0.61
Non-Hamiltonian, N = 12	25.6	11	80	0.53
50 graphs, N = 20	29.5	1	141	1.08
50 graphs, N = 30	216.5	3	1057	10.41
50 graphs, N = 40	2595.6	52	10536	160.09
50 graphs, N = 50	40316.7	324	232812	2171.46

generated graphs are Hamiltonian. See Meringer [79] for a reference on generating cubic graphs. We run every test with $\beta = 0.99$ and $\mu = 1/N^2$.

With this basic implementation of the B&F, we are not able to obtain a Hamiltonian solution for the 8×8 Knight's Tour problem after 12 hours. In Section 7.4, however, we introduce new constraints that allow us solve this problem in little more than a minute.

Example 7.1 *We describe a solution of the envelope graph (Figure 7.1), obtained using the aforementioned implementation of the B&F method, with* $\beta = 0.99$ *and* $\mu = 1/36$. *First, we solve the feasibility problem*

$$\begin{aligned}
x_{12} + x_{14} + x_{15} - \beta x_{21} - \beta x_{41} - \beta x_{51} &= 1 - 5\mu, \\
x_{21} + x_{23} + x_{26} - \beta x_{12} - \beta x_{32} - \beta x_{62} &= \mu, \\
x_{32} + x_{34} + x_{36} - \beta x_{23} - \beta x_{43} - \beta x_{63} &= \mu, \\
x_{41} + x_{43} + x_{45} - \beta x_{14} - \beta x_{34} - \beta x_{54} &= \mu, \\
x_{51} + x_{54} + x_{56} - \beta x_{15} - \beta x_{45} - \beta x_{65} &= \mu, \\
x_{62} + x_{63} + x_{65} - \beta x_{26} - \beta x_{36} - \beta x_{56} &= \mu, \\
x_{12} + x_{14} + x_{15} &= \frac{(1-5\mu)(1-\beta) + \mu(\beta - \beta^6)}{(1-\beta)(1-\beta^6)}, \\
x_{ia} &\geq 0, \qquad \textit{for all } (i,a) \in E(G).
\end{aligned}$$

The first iteration produces a 1-randomised policy where the randomisation occurs at vertex 4. The logical B&F tree then splits into three choices: to fix arc (4,1), (4,3) or (4,5).

The algorithm first branches on fixing arc (4,1) (Figure 7.3). As the algorithm uses a depth-first search, arcs (4,3) and (4,5) will not be fixed unless the algorithm fathoms the (4,1) branch without having found a Hamiltonian cycle. Fixing the arc (4,1) is equivalent to eliminating arcs (4,3) and (4,5) in the remainder of this branch of the logical B&F tree. In addition, arcs (1,4),

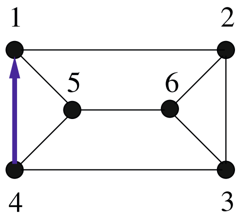

Fig. 7.3: Branching on arc $(4,1)$

(2,1) and (5,1) can also be eliminated because they cannot be present together with arc (4,1) in a Hamiltonian cycle.

At the second iteration we solve two LPs. We first solve the second LP, to check the feasibility of the graph remaining after the above round of fixing (and eliminating) of arcs

$$\min \{x_{12} + x_{15}\} \tag{7.11}$$

subject to

$$\begin{aligned}
x_{12} + x_{15} - \beta x_{41} &= 1 - 5\mu, \\
x_{23} + x_{26} - \beta x_{12} - \beta x_{32} - \beta x_{62} &= \mu, \\
x_{32} + x_{34} + x_{36} - \beta x_{23} - \beta x_{63} &= \mu, \\
x_{41} - \beta x_{34} - \beta x_{54} &= \mu, \\
x_{54} + x_{56} - \beta x_{15} - \beta x_{65} &= \mu, \\
x_{62} + x_{63} + x_{65} - \beta x_{26} - \beta x_{36} - \beta x_{56} &= \mu, \\
-\beta^6 x_{12} - \beta^6 x_{15} + \beta x_{41} &= \frac{\mu(\beta - \beta^6)}{1 - \beta}, \\
x_{ia} &\geq 0, \quad \textit{for all } (i,a) \in E(G).
\end{aligned}$$

The last equality constraint above comes from (7.6) because the fixed arc $(4,1)$ *returns to the home vertex. The optimal objective function returned is equal to the right-hand side of the omitted constraint (4.19), so we cannot fathom this branch, at this stage. Hence, we also solve the updated LP (7.1)*

$$\min \{-\beta^6 x_{12} - \beta^6 x_{15} + \beta x_{41}\}$$

subject to

$$\begin{aligned}
x_{12} + x_{15} - \beta x_{41} &= 1 - 5\mu, \\
x_{23} + x_{26} - \beta x_{12} - \beta x_{32} - \beta x_{62} &= \mu, \\
x_{32} + x_{34} + x_{36} - \beta x_{23} - \beta x_{63} &= \mu,
\end{aligned}$$

$$
\begin{aligned}
x_{41} - \beta x_{34} - \beta x_{54} &= \mu, \\
x_{54} + x_{56} - \beta x_{15} - \beta x_{65} &= \mu, \\
x_{62} + x_{63} + x_{65} - \beta x_{26} - \beta x_{36} - \beta x_{56} &= \mu, \\
x_{12} + x_{15} &= \frac{(1-5\mu)(1-\beta) + \mu(\beta - \beta^6)}{(1-\beta)(1-\beta^6)}, \\
x_{ia} &\geq 0, \quad \textit{for all } (i,a) \in E(G).
\end{aligned}
$$

The second iteration produces a 1-randomised policy where the randomisation occurs at vertex 3. The logical B&F tree then splits into three choices: to fix arc (3,2), (3,4) or (3,6). The algorithm first selects the arc (3,2) to continue the branch. The graph at this stage is shown in Figure 7.4.

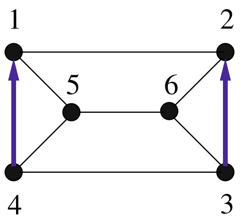

Fig. 7.4: Branching on arc $(3,2)$ after fixing arc $(4,1)$

Applying Checks 1–4 to the remaining vertices with multiple non-fixed arcs, we immediately see that arcs (2,6) and (1,5) must be fixed by Check 4. Once these arcs are fixed, arcs (5,4) and (6,3) are also fixed by Check 4. At this stage, every vertex has a fixed arc but we have not obtained a Hamiltonian cycle; hence, we fathom the branch. Travelling back up the tree, the algorithm next selects the arc (3,4) to branch on. Figure 7.5 shows the current graph.

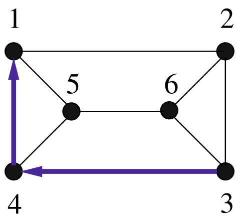

Fig. 7.5: Second branching on arc $(3,4)$

Applying Checks 1–4 to the remaining vertices with multiple non-fixed arcs, we immediately see that arc (5,6) must be fixed by Check 3. Once this arc is fixed, arc (2,3) is also fixed by Check 3. Next, arc (6,2) is fixed by

Check 4 and, finally, arc (1,5) is fixed by Check 3. At this stage, every vertex has a fixed arc. Since these fixed arcs correspond to the Hamiltonian cycle $1 \to 5 \to 6 \to 2 \to 3 \to 4 \to 1$*, the algorithm terminates. The Hamiltonian cycle is shown in Figure 7.6.*

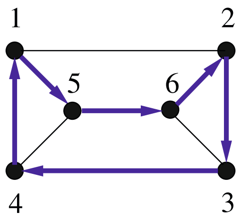

Fig. 7.6: Hamiltonian cycle found by the B&F method

The whole logical B&F tree is illustrated in Figure 7.7.

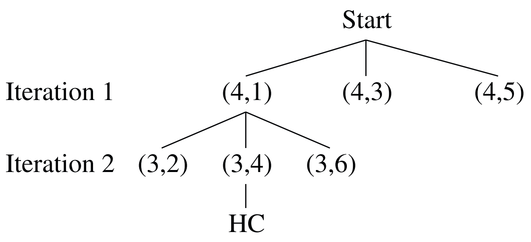

Fig. 7.7: The logical B&F tree for the envelope graph

We note that even in this simple example, in the worst case $3^6 = 729$ *branches are generated. However, our algorithm is able to find a Hamiltonian cycle after examining only two.*

Branching methods

A standard question when using a branching algorithm is which method of branching to use. A major benefit of the branch and fix method is that the checks and the second LP (7.9) often allow us to fathom a branching point relatively early, so depth-first searching is used. However, the horizontal ordering of the branch tree is, by default, determined by nothing more than the initial ordering of the vertices. For a non-Hamiltonian graph, this ordering makes no difference as the breadth of the entire tree will need to be traversed to determine that no Hamiltonian cycles exist. However, for a Hamiltonian graph, it is possible that a relabelling of the graph would result in the branch and fix method finding a Hamiltonian cycle sooner.

While it seems impossible to predict, in advance, which relabelling of vertices will find a Hamiltonian cycle the quickest, it is possible that the structure of the 1-randomised policy found at each branching point can provide information about which branch should be traversed first. Each 1-randomised policy with the splitting vertex i contains two non-zero values x_{ij} and x_{ik}, $j \neq k$. Without loss of generality, assume that $x_{ij} \leq x_{ik}$. We propose five branching methods:

1. Default branching (or vertex order): the branches are traversed in the order of the numerical labels of the vertices.

2. First branch on fixing (i,j), then (i,k) and then traverse the rest of the branches in vertex order.

3. First branch on fixing (i,k), then (i,j) and then traverse the rest of the branches in vertex order.

4. All branches are traversed in vertex order other than those corresponding to fixing (i,j) or (i,k). The last two branches traversed are (i,j) and then (i,k).

5. First branch on fixing (i,k), then traverse the rest of the branches other than the branch corresponding to fixing (i,j) in vertex order, and finally branch on fixing (i,j).

We test these five branching methods on the same sets of 50 randomly generated Hamiltonian cubic graphs as those generated for Table 7.2. In Table 7.3, we give the average number of branches examined by the B&F for each branching method.

Table 7.3: Average number of branches examined by the five branching methods over sets of Hamiltonian cubic graphs

Branching method	**20-vertex graphs**	**30-vertex graphs**	**40-vertex graphs**	**50-vertex graphs**
1	29.48	216.54	2595.58	40316.68
2	33.28	261.24	2227.24	43646.92
3	24.58	172.30	1624.50	17468.26
4	38.68	285.26	2834.44	53719.96
5	34.04	345.44	3228.44	76159.30

From the results shown in Table 7.3, it appears that Branching Method 3 is the best performing method for cubic graphs. The sets of cubic graphs are produced by GENREG [79], which uses a particular strategy of ordering the vertices that may also account for the success of Branching Method 3.

7.4 Wedge Constraints

Recall that constraints (4.18)–(4.20) that define $\widetilde{\mathcal{X}}_\beta$ depend upon parameters β and μ. While the use of β is necessary in the framework of a discounted Markov decision process, the selection of μ as a small, positive parameter is used only to ensure that the mapping $\boldsymbol{f}_x(i,a) = x_{ia}/x_i$ (see (4.17)) is well-defined. Without setting $\mu > 0$, it is possible for x_i to be 0. To illustrate this, recall constraint (4.18)

$$\sum_{i=1}^{N} \sum_{a \in \mathcal{A}(i)} \left(\delta_{ij} - \beta p(j|i,a)\right) x_{ia} = \nu_j, \quad \text{for } j \in \mathcal{S},$$

where

$$\nu_j = \begin{cases} 1 - (N-1)\mu, & \text{if } i = 1, \\ \mu, & \text{otherwise.} \end{cases}$$

Rearranging constraint (4.18), we obtain

$$\sum_{a \in \mathcal{A}(j)} x_{ja} = \beta \sum_{i=1}^{N} \sum_{a \in \mathcal{A}(i)} p(j|i,a) x_{ia} + \nu_j, \quad \text{for } j \in \mathcal{S}.$$

Since we cannot ensure that

$$\sum_{i=1}^{N} \sum_{a \in \mathcal{A}(i)} p(j|i,a) x_{ia} \neq 0$$

for all j, we select $\mu > 0$ to ensure that $x_j = \sum_{a \in \mathcal{A}(j)} x_{ja} > 0$. However, an additional set of constraints, introduced in Eshragh *et al.* [43], can achieve the same goal while allowing us to set $\mu = 0$ by bounding x_j away from 0. We call these constraints *wedge constraints*. The wedge constraints are comprised of the following two sets of inequalities:

$$\sum_{a \in \mathcal{A}(i)} x_{ia} \leq \frac{\beta}{1 - \beta^N}, \quad \text{for } i = 2, \ldots, N, \tag{7.12}$$

$$\sum_{a \in \mathcal{A}(i)} x_{ia} \geq \frac{\beta^{N-1}}{1 - \beta^N}, \quad \text{for } i = 2, \ldots, N. \tag{7.13}$$

The rationale behind the wedge constraints is that when $\mu = 0$, we know from (7.2) that all Hamiltonian solutions to (4.18)–(4.20) take the form

$$x_{ia} = \begin{cases} \beta^k/(1-\beta^N) & \text{if } (i,a) \text{ is the } k\text{th arc on the HC,} \\ 0 & \text{otherwise.} \end{cases} \tag{7.14}$$

In every Hamiltonian cycle, exactly one arc emanates from each vertex. If (i_k, i_{k+1}) is an arc on a Hamiltonian cycle, then,

$$\sum_{a \in \mathcal{A}(i_k)} x_{i_k a} = x_{i_k i_{k+1}}. \tag{7.15}$$

Recall that we define the home vertex of a graph as vertex 1. Then, the initial (0th) arc in any Hamiltonian cycle is arc $(1, a)$, for some $a \in \mathcal{A}(1)$; therefore, from (7.14) we obtain

$$\sum_{a \in \mathcal{A}(1)} x_{1a} = \frac{1}{1-\beta^N}. \tag{7.16}$$

Constraint (7.16) is already given in (4.19) if we set $\mu = 0$. For all other vertices, however, constraints (7.12)–(7.13) will partially capture a new property of Hamiltonian solutions that is expressed in (7.14). In particular, substituting (7.14) into (7.15), for all vertices other than the home vertex, we obtain wedge constraints (7.12)–(7.13). Recall that in a cubic graph, there are exactly three arcs—say (i, a), (i, b) and (i, c)—from a given vertex i. Thus, in 3-dimensions, the corresponding constraints (7.12)–(7.13) have the shape indicated in Figure 7.8 that looks like a slice of a pyramid. The resulting wedge-like shape inspires the name wedge constraints.

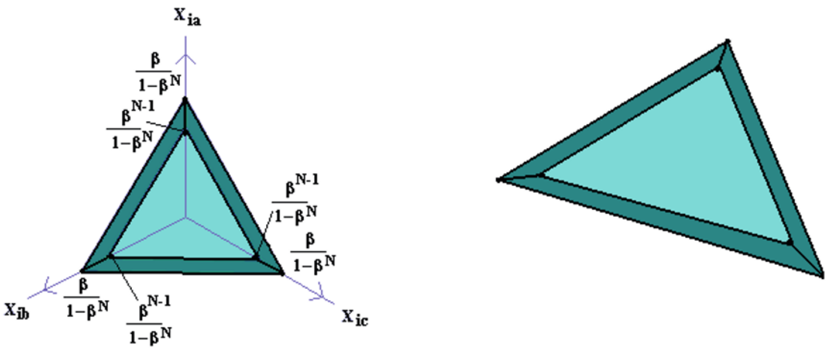
x_{ia}
$\frac{\beta}{1-\beta^N}$
$\frac{\beta^{N-1}}{1-\beta^N}$
$\frac{\beta^{N-1}}{1-\beta^N}$
$\frac{\beta}{1-\beta^N}$
x_{ib}
$\frac{\beta}{1-\beta^N}$
$\frac{\beta^{N-1}}{1-\beta^N}$
x_{ic}

Fig. 7.8: Wedge constraints for a vertex in a cubic graph

We can add wedge constraints (7.12)–(7.13) to the constraint set (4.18)–(4.20), setting $\mu = 0$ in the latter. However, adding wedge constraints destroys the 1-randomised structure of non-Hamiltonian solutions that exists for the extreme points of the feasible region specified by (4.18)–(4.20), introducing many new extreme points to the feasible region. Since this is undesirable, we only use wedge constraints when solving the second LP (with the objective function (7.9)), in an attempt to determine whether a branch can be fathomed earlier than is the case without the wedge constraints.

The model incorporating the wedge constraints is

$$\sum_{i=1}^{N} \sum_{a \in \mathcal{A}(i)} (\delta_{ij} - \beta p(j|i,a))\, x_{ia} = \delta_{1j}, \quad \text{for } j \in \mathcal{S}, \tag{7.17}$$

$$x_{ia} \geq 0, \quad \text{for } (i,a) \in E(G), \tag{7.18}$$

$$\sum_{a \in \mathcal{A}(i)} x_{ia} \leq \frac{\beta}{1-\beta^N}, \quad \text{for } i = 2, \ldots, N, \tag{7.19}$$

$$\sum_{a \in \mathcal{A}(i)} x_{ia} \geq \frac{\beta^{N-1}}{1-\beta^N}, \quad \text{for } i = 2, \ldots, N, \tag{7.20}$$

which replaces constraints (4.18) and (4.20) in the second LP (7.9). Everything else in the branch and fix method is identical to the method described in Section 7.2, except that the obtained value of the objective function (7.9) is now compared to the right-hand side of (7.16), that is, the right-hand side of (4.19) with μ set to 0.

We run this model on the same selection of graphs as those in Tables 7.1–7.3, to compare its performance to that of the original branch and fix method. There is a significant decrease in the number of branches examined, and consequently in the running time of the model. Tables 7.4–7.6 show a sample of results. We run all tests with $\beta = 0.99$, and $\mu = 1/N^2$. In Table 7.4, we compare the number of branches examined by the B&F for five graphs to the maximum possible number of branches (deterministic policies), and show the running time in seconds. In Table 7.5, we solve several sets of cubic graphs, and report the average number of branches examined by the B&F over each set. We also report the minimum and maximum branches examined over each set, and the average running time. As in Table 7.2, we consider all 10-vertex cubic graphs, of which there are 17 Hamiltonian and 2 non-Hamiltonian graphs, and all 12-vertex cubic graphs, of which there are 80 Hamiltonian and 5 non-Hamiltonian graphs. For each set of larger cubic graphs, we use the same graphs as were randomly generated for Table 7.2. Table 7.6 shows the average number of branches examined by the B&F for the same set of randomly cubic graphs shown in Table 7.5, for all five branching methods.

Table 7.4: Preliminary results for the branch and fix method with wedge constraints included

Graph	**Branches**	**Upper bound**	**Time(s)**
Dodecahedron: Ham, $N = 20$, arcs = 60	43	3.4868×10^{9}	1.71
Ham, $N = 24$, arcs = 72	5	2.8243×10^{11}	0.39
8×8 Knight's Tour: Ham, $N = 64$, arcs = 336	220	9.1654×10^{43}	78.46
Petersen: non-Ham, $N = 10$, arcs = 30	53	5.9049×10^{4}	1.17
Coxeter: non-Ham, $N = 28$, arcs = 84	5126	2.2877×10^{13}	262.28

Table 7.5: Performance of the branch and fix method with wedge constraints included over cubic graphs

Type of graphs	**Average branches**	**Minimum branches**	**Maximum branches**	**Average time(s)**
Hamiltonian, N = 10	2.1	1	4	0.09
Hamiltonian, N = 12	3.0	1	10	0.14
Non-Hamiltonian, N = 10	32.5	12	53	0.70
Non-Hamiltonian, N = 12	23.2	11	72	0.50
50 graphs, N = 20	14.6	1	75	0.65
50 graphs, N = 30	41.7	2	182	2.56
50 graphs, N = 40	209.2	7	1264	18.11
50 graphs, N = 50	584.4	8	2522	67.99

Table 7.6: Average number of branches examined by the five branching methods with wedge constraints included over sets of Hamiltonian cubic graphs

Branching method	**20-vertex graphs**	**30-vertex graphs**	**40-vertex graphs**	**50-vertex graphs**
1	14.60	41.66	209.18	584.40
2	14.76	49.94	164.74	636.88
3	11.86	38	145.54	603.24
4	17.28	50.32	152.50	632.08
5	15.32	39.42	123.72	356.32

We observe that for smaller-size graphs, wedge constraints do not improve the performance of the B&F method. For larger-size graphs, however, a significant improvement is evident. This is especially striking for graphs with 30–40 vertices. We also note with interest that the model with wedge constraints included performs well when we select Branching Method 5, which is the worst performing branching method in the model without wedge constraints. Since Branching Method 3 also performs well, and both Branching Methods 3 and 5 involve branching first on arc (i, k), it appears that branching first on arc (i, k) is an efficient strategy for graphs generated by GENREG when we include wedge constraints.

In Andramonov *et al.* [4], the first numerical procedure taking advantage of the MDP embedding is used to solve graphs of similar sizes to those listed in Tables 7.4 and 7.5. The model given in [4] is solved using the Mix Integer Programming (MIP) solver in CPLEX 4.0. The present model outperforms the model in [4] in terms of the number of branches examined in the cases displayed in Tables 7.4 and 7.5. In particular, the number of branches examined to solve each graph is much reduced in the branch and fix method when compared to similar sized graphs in [4]. In particular, to solve the 8×8 Knight's Tour graph, between 2000 and 40000 branches were examined in the model given in [4] (depending on the selection of parameters in CPLEX), but only 220 were required in the branch and fix method with wedge constraints included. This significant improvement highlights both the progress made in this line of research over the last decade, and the advantages obtained by the use of wedge constraints. We seek to take further advantage of the wedge constraints in a new mixed integer programming formulation, *the Wedged-MIP heuristic*, which we describe in the next section.

7.5 The Wedged-MIP heuristic

The discussion in the preceding section naturally leads us to consider the polytope $\mathcal{Y}_\beta$ defined by the following seven sets of linear constraints

$$\sum_{i=1}^{N} \sum_{a \in \mathcal{A}(i)} (\delta_{ij} - \beta p(j|i,a))\, y_{ia} = \delta_{1j}(1 - \beta^N), \quad \text{for } j \in \mathcal{S}, \tag{7.21}$$

$$\sum_{a \in \mathcal{A}(1)} y_{1a} = 1, \tag{7.22}$$

$$y_{ia} \geq 0, \quad \text{for } (i,a) \in E(G), \tag{7.23}$$

$$\sum_{j \in \mathcal{A}(i)} y_{ij} \leq \beta, \quad \text{for } i \in \mathcal{S}\backslash\{1\}, \tag{7.24}$$

$$\sum_{j \in \mathcal{A}(i)} y_{ij} \geq \beta^{N-1}, \quad \text{for } i \in \mathcal{S}\backslash\{1\}, \tag{7.25}$$

$$y_{ij} + y_{ji} \leq 1, \quad \text{for } (i,j),(j,k) \in E(G), \tag{7.26}$$

$$y_{ij} + y_{jk} + y_{ki} \leq 2, \quad \text{for } (i,j),(j,k),(k,i) \in E(G). \tag{7.27}$$

Remark 7.1.

1. The variables y_{ia} that define $\mathcal{Y}_\beta$ are obtained from the variables x_{ia} that define $\mathcal{X}_\beta$ by the transformation

$$y_{ia} = (1 - \beta^N) x_{ia}, \quad \text{for } (i,a) \in E(G).$$

2. In view of Item 1, constraints (7.21)–(7.23) are merely constraints (4.18)–(4.20) normalised by the multiplier $(1 - \beta^N)$, and with $\mu = 0$. Furthermore, constraints (7.25)–(7.24) are similarly normalised wedge constraints (7.12)–(7.13).

3. Note that normalising (7.14) in the same way implies that if $\boldsymbol{y} \in \mathcal{Y}_\beta$ corresponds to a Hamiltonian cycle, then

$$y_{ia} \in \{0, 1, \beta, \ldots, \beta^{N-1}\}, \qquad \text{for } (i,a) \in E(G).$$

   Thus, for β sufficiently near 1 all positive entries of a Hamiltonian solution $\boldsymbol{y}$ are also either 1 (if $i = 1$), or close to 1. Therefore, if $\boldsymbol{y} \in \mathcal{Y}_\beta$ is any feasible point with only one positive entry y_{ia} for all $a \in \mathcal{A}(i)$, for each i, then constraints (7.25)–(7.24) ensure that all those positive entries have values near 1. Furthermore, constraints (7.26)–(7.27) ensure that at most one such large entry is permitted on any potential 2-cycle, and at most two such large entries are permitted on any potential 3-cycle.

4. In view of Item 3, it is reasonable to search for a feasible point $\boldsymbol{y} \in \mathcal{Y}_\beta$ that has only a single positive entry y_{ia} for all $a \in \mathcal{A}(i)$ and for each i.

We make the last point of the above remark precise in the following proposition that is analogous to a result proved in Chen and Filar [22] for an embedding of the HCP in a long-run average MDP. Since it forms the theoretical basis of our most powerful heuristic, for the sake of completeness, we supply a formal proof below.

Proposition 7.1. *Given any graph G and its embedding in a discounted Markov decision process, consider the polytope $\mathcal{Y}_\beta$ defined by (7.21)–(7.27) for $\beta \in [0,1)$ and sufficiently near 1. The following statements are equivalent:*

(i) The point $\hat{\boldsymbol{y}} \in \mathcal{Y}_\beta$ is Hamiltonian in the sense that the positive entries $\hat{y}_{ia}$ of $\hat{\boldsymbol{y}}$ correspond to arcs (i,a) defining a Hamiltonian cycle in G.

(ii) The point $\hat{\boldsymbol{y}} \in \mathcal{Y}_\beta$ is a global minimiser of the nonlinear program

$$\begin{aligned} &\min \sum_{i=1}^{N} \sum_{a \in \mathcal{A}(i)} \sum_{b \in \mathcal{A}(i), b \neq a} y_{ia} y_{ib} \\ &\textit{subject to} \qquad\qquad\qquad\qquad (7.28) \\ &\boldsymbol{y} \in \mathcal{Y}_\beta. \end{aligned}$$

which gives the objective function value of 0 in (7.28).

(iii) The point $\hat{\boldsymbol{y}} \in \mathcal{Y}_\beta$ satisfies the additional set of nonlinear constraints

$$y_{ia} y_{ib} = 0, \quad \textit{for } i \in \mathcal{S}, a, b \in \mathcal{A}(i), a \neq b, \qquad (7.29)$$

Proof. By the nonnegativity of $\hat{\boldsymbol{y}} \in \mathcal{Y}_\beta$, it immediately follows that Parts (ii) and (iii) are equivalent.

From (7.14) and Item 3 of Remark 7.1 we note that if $\hat{\boldsymbol{y}}$ is Hamiltonian, it implies Part (iii). Furthermore, if $\hat{\boldsymbol{y}}$ is a global minimiser of (7.28), constraints (7.25) ensure that it must have at least one positive entry corresponding to some arc $a \in \mathcal{A}(i)$ for each $i \in \mathcal{S}$. Since $\hat{y}_{ia}\hat{y}_{ib} = 0$ for all $a \neq b$, $i \in \mathcal{S}$, we conclude that $\hat{\boldsymbol{y}}$ has exactly one positive entry $\hat{y}_{ia}$, for each i. We then define $\hat{\boldsymbol{x}} \in \widetilde{\mathcal{X}}_\beta$ by $\hat{x}_{ia} = 1/(1-\beta^N)\hat{y}_{ia}$ for all $(i,a) \in E(G)$, and use (7.14) to construct the policy $\hat{\boldsymbol{f}} = M^{-1}(\hat{\boldsymbol{x}})$. It is clear that $\hat{\boldsymbol{f}} \in \mathcal{F}_\mathcal{D}$, and hence $\hat{\boldsymbol{x}}$ is Hamiltonian by Part (ii) of Proposition 4.2. Since positive entries of $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ occur at precisely the same arcs (i,a), $\hat{\boldsymbol{y}}$ is also Hamiltonian, so Part (iii) implies Part (i). □

Corollary 7.2. *If $\mathcal{Y}_\beta = \emptyset$, the empty set, then the graph G is non-Hamiltonian. If $\mathcal{Y}_\beta \neq \emptyset$, the (possibly empty) set of Hamiltonian solutions $\mathcal{Y}_\beta^{\mathrm{H}} \subset \mathcal{Y}_\beta$ is in one-to-one correspondence with Hamiltonian cycles of G, and satisfies*

$$\mathcal{Y}_\beta^{\mathrm{H}} = \mathcal{Y}_\beta \cap \{\boldsymbol{y} \mid (7.29) \textit{ holds}\}.$$

Proof. By construction, if G is Hamiltonian, then there exists a policy $\hat{\boldsymbol{f}} \in \mathcal{F}_\mathcal{D}$ tracing out a Hamiltonian cycle in G. Let $\hat{\boldsymbol{x}} = M(\hat{\boldsymbol{f}})$ using (4.13) with $\boldsymbol{\nu} = \boldsymbol{e}_1^{\mathrm{T}}$, and define $\hat{\boldsymbol{y}} = (1-\beta^N)\hat{\boldsymbol{x}}$. Clearly, $\hat{\boldsymbol{y}} \in \mathcal{Y}_\beta$, so $\mathcal{Y}_\beta \neq \emptyset$. From Proposition 7.1, it follows that $\hat{\boldsymbol{y}}$ satisfies (7.29). Conversely, only points in $\mathcal{Y}_\beta^{\mathrm{H}}$ define Hamiltonian solutions in $\mathcal{Y}_\beta$. □

For symmetric graphs, in which arc $(i,a) \in E(G)$ if and only if arc $(a,i) \in E(G)$, we further improve the wedge constraints (7.25)–(7.24) by considering the shortest path between the home vertex and each other vertex in the graph. We define $\ell(i,j)$ to be the length of the shortest path between vertices i and j in G.

Lemma 7.1. *Any Hamiltonian solution to (7.21)–(7.23) satisfies the following constraints*

$$\sum_{a \in \mathcal{A}(i)} y_{ia} \leq \beta^{\ell(1,i)}, \quad \textit{for } i \in \mathcal{S}\backslash\{1\}, \tag{7.30}$$

$$\sum_{a \in \mathcal{A}(i)} y_{ia} \geq \beta^{N-\ell(1,i)}, \quad \textit{for } i \in \mathcal{S}\backslash\{1\}. \tag{7.31}$$

Proof. From (7.14) and Item 3 of Remark 7.1, we know that for a Hamiltonian cycle in which the kth arc is the arc (i,a), the corresponding variable $y_{ia} = \beta^k$, and all other $y_{ib} = 0$, $b \neq a$. Therefore,

$$\sum_{a \in \mathcal{A}(i)} y_{ia} = \beta^k. \tag{7.32}$$

Then, since it takes at least $\ell(1,i)$ arcs to reach vertex i from the home vertex 1, we immediately obtain that $k \geq \ell(1,i)$, and therefore

$$\sum_{a\in\mathcal{A}(i)} y_{ia} \leq \beta^{\ell(1,i)},$$

which coincides with (7.30). Since G is an undirected graph, we know that $\ell(i,1) = \ell(1,i)$. Thus, we obtain that $k \leq N - \ell(1,i)$, and hence,

$$\sum_{a\in\mathcal{A}(i)} y_{ia} \geq \beta^{N-\ell(1,i)},$$

which coincides with (7.31). □

Given the above, we reformulate the HCP as a mixed (non-linear) integer programming feasibility problem, which we call the *Wedged-MIP heuristic*, as follows

$$\sum_{i=1}^{N}\sum_{a\in\mathcal{A}(i)} \left(\delta_{ij} - \beta p(j|i,a)\right) y_{ia} = \delta_{1j}(1-\beta^N), \quad \text{for } j \in \mathcal{S}, \tag{7.33}$$

$$\sum_{a\in\mathcal{A}(1)} y_{1a} = 1, \tag{7.34}$$

$$y_{ia} \geq 0, \quad \text{for } (i,a) \in E(G), \tag{7.35}$$

$$\sum_{j\in\mathcal{A}(i)} y_{ij} \leq \beta^{\ell(1,i)}, \quad \text{for } i \in \mathcal{S}\backslash\{1\}, \tag{7.36}$$

$$\sum_{j\in\mathcal{A}(i)} y_{ij} \geq \beta^{N-\ell(1,i)}, \quad \text{for } i \in \mathcal{S}\backslash\{1\}, \tag{7.37}$$

$$y_{ij} + y_{ji} \leq 1, \quad \text{for } (i,j),(j,i) \in E(G), \tag{7.38}$$

$$y_{ij} + y_{jk} + y_{ki} \leq 2, \quad \text{for } (i,j),(j,k),(k,i) \in E(G), \tag{7.39}$$

$$y_{ia}y_{ib} = 0, \quad \text{for } i \in \mathcal{S}, a,b \in \mathcal{A}(i), a \neq b. \tag{7.40}$$

We solve the above formulation in IBM ILOG OPL-CPLEX 5.1. A benefit of this solver is that the non-linear constraints (7.40) may be submitted in a format usually not acceptable in CPLEX and the IBM ILOG OPL-CPLEX CP Optimizer will interpret them in a way suitable for CPLEX. We allow these constraints to be submitted in one of two different ways, left up to the user's choice. We define the operator == as follows

$$(a == b) = \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{otherwise,} \end{cases}$$

the operator $!=$ as follows

$$(a \; != b) = \begin{cases} 0 & \text{if } a = b, \\ 1 & \text{otherwise,} \end{cases}$$

and denote by d_i the number of arcs emanating from vertex i. Then, we submit constraints (7.40) to IBM ILOG OPL-CPLEX in either of the forms

$$\sum_{a \in \mathcal{A}(i)} (y_{ia} == 0) = d_i - 1, \quad \text{for } i \in \mathcal{S}, \tag{7.41}$$

or

$$\sum_{a \in \mathcal{A}(i)} (y_{ia} \; != 0) = 1, \quad \text{for } i \in \mathcal{S}. \tag{7.42}$$

Even though constraints (7.41) and (7.42) are theoretically identical when added to (7.33)–(7.39), their interpretation by IBM ILOG OPL-CPLEX produces different solutions, with different running times. Neither choice solves graphs consistently faster than the other, so if one form fails to find a solution quickly, we try the other form. Using this model we are able to obtain Hamiltonian solutions efficiently for many large graphs, using a Pentium 3.4GHz with 4GB RAM. Table 7.7 presents results for eight graphs.

Table 7.7: Running times for Wedged-MIP heuristic

Graph	N	**Arcs**	β	**Time(hh:mm:ss)**
8×8 Knight's Tour	64	336	0.99999	00:00:02
Perturbed Horton	94	282	0.99999	00:00:02
12×12 Knight's Tour	144	880	0.99999	00:00:03
250-vertex	250	1128	0.99999	00:00:16
20×20 Knight's Tour	400	2736	0.99999	00:20:57
500-vertex	500	3046	0.99999	00:10:01
1000-vertex	1000	3996	0.999999	00:30:46
2000-vertex	2000	7992	0.999999	10:24:05

We note that the perturbed Horton graph given here is a 94-vertex cubic graph that, unlike the original Horton graph (96-vertex cubic graph [103]), is Hamiltonian. The 250- and 500-vertex graphs are both non-regular graphs that are randomly generated for testing purposes, while the 1000- and 2000-vertex graphs come from the TSPLIB website, maintained by University of Heidelberg [98].

We present a visual representation of a solution to the 250-vertex graph found by the Wedged-MIP heuristic in Figure 7.9, where the vertices are drawn as blue dots clockwise in an ellipse, with vertex 1 at the top, and the arcs between the vertices are inside the ellipse. The arcs in the Hamiltonian cycle found by the Wedged-MIP heuristic are highlighted red, and all other arcs are shown in blue. While it is very difficult to make out much detail from Figure 7.9, it serves as a good illustration of the complexity involved in problems of this size.

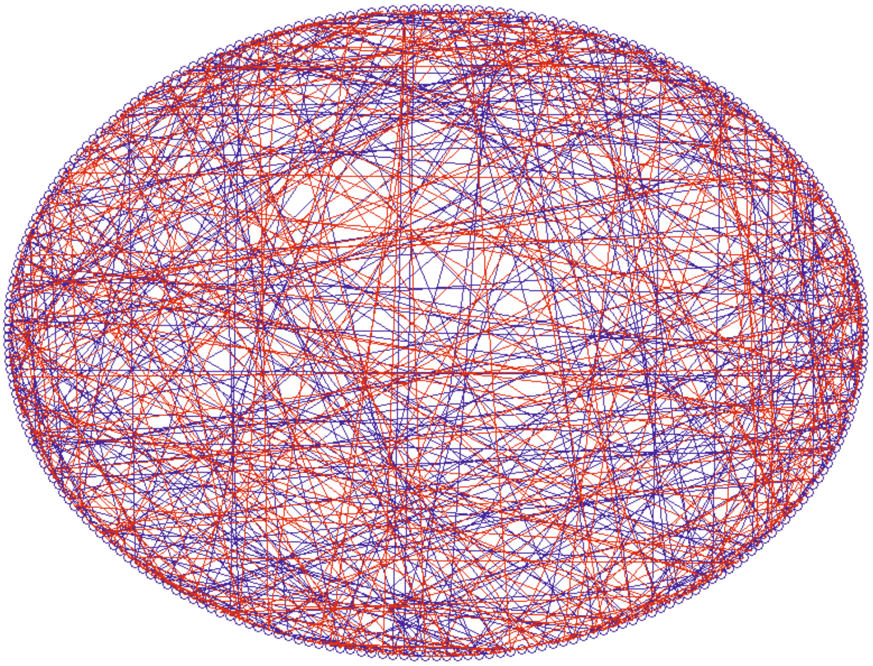

Fig. 7.9: Solution to 250-vertex graph (Hamiltonian cycle in red)

Comparisons between Wedged-MIP heuristic and two TSP formulations

Since we solve the Wedged-MIP heuristic in OPL-CPLEX, we investigate two other, well-known, MIP formulations, and also solve them in OPL-CPLEX for the same set of graphs as those in Table 7.7, as well as three randomly generated cubic Hamiltonian graphs of sizes 12, 24 and 38, as a benchmark test. The two other formulations are the *modified single commodity flow model* [54] and the *third stage dependent model* [101]. We select these two formulations because they are the best performing methods of each type (commodity flow and stage dependent, respectively) reported in Orman and Williams [83]. Both the modified single commodity flow and the third stage dependent models were designed to solve the travelling salesman problem, so they contain costs/distances c_{ij} for each arc $(i, j) \in E(G)$. Since we only want to solve the HCP with these formulations, we set

$$c_{ij} = \begin{cases} 1 & \text{for } (i,j) \in E(G), \\ 0 & \text{for } (i,j) \notin E(G). \end{cases}$$

The modified single commodity flow model (MSCF) is formulated with decision variables x_{ij} and y_{ij} as follows

$$\min \sum_{(i,j)\in E(G)} c_{ij}x_{ij}$$

subject to

$$\begin{aligned}
\sum_{j\in\mathcal{A}(i)} x_{ij} &= 1, \quad \text{for } i \in \mathcal{S}, \\
\sum_{i\in\mathcal{B}(j)} x_{ij} &= 1, \quad \text{for } j \in \mathcal{S}, \\
\sum_{j\in\mathcal{A}(1)} y_{1j} &= N-1, \\
\sum_{i\in\mathcal{B}(j)} y_{ij} - \sum_{k\in\mathcal{A}(j)} y_{jk} &= 1, \quad \text{for } j \in \mathcal{S}\backslash\{1\}, \\
y_{1j} &\le (N-1)x_{1j}, \quad \text{for } j \in \mathcal{A}(1), \\
y_{ij} &\le (N-2)x_{ij}, \quad \text{for } i \in \mathcal{S}\backslash\{1\}, j \in \mathcal{A}(i), \\
x_{ij} &\in \{0,1\}, \\
y_{ij} &\ge 0.
\end{aligned}$$

The third stage dependent model (TSD) is formulated with decision variables x_{ij} and $y^t_{ij}, t \in \mathcal{S}$, as follows

$$\min \sum_{(i,j)\in E(G)} c_{ij}x_{ij}$$

subject to

$$\begin{aligned}
\sum_{j\in\mathcal{A}(i)} x_{ij} &= 1, \quad \text{for } i \in \mathcal{S}, \\
\sum_{i\in\mathcal{B}(j)} x_{ij} &= 1, \quad \text{for } j \in \mathcal{S},
\end{aligned}$$

$$\begin{aligned}
x_{ij} - \sum_{t=1}^{N} y_{ij}^{t} &= 0, \quad \text{for } (i,j) \in E(G),\\
\sum_{t=1}^{N} \sum_{j \in \mathcal{A}(i)} y_{ij}^{t} &= 1, \quad \text{for } i \in \mathcal{S},\\
\sum_{t=1}^{N} \sum_{i \in \mathcal{B}(j)} y_{ij}^{t} &= 1, \quad \text{for } j \in \mathcal{S},\\
\sum_{(i,j) \in E(G)} y_{ij}^{t} &= 1, \quad \text{for } t \in \mathcal{S},\\
\sum_{j \in \mathcal{A}(1)} y_{1j}^{1} &= 1,\\
\sum_{i \in \mathcal{B}(1)} y_{i1}^{N} &= 1,\\
\sum_{j \in \mathcal{A}(i)} y_{ij}^{t} - \sum_{k \in \mathcal{B}(i)} y_{ki}^{t-1} &= 0, \quad \text{for } i \in \mathcal{S}, \text{ and } t \in \mathcal{S}\backslash\{1\}.
\end{aligned}$$

We run these two models on the same graphs as shown in Table 7.7, as well as three additional randomly generated cubic Hamiltonian graphs of orders 12, 24 and 38. Table 7.8 shows the running times of these two models, along with the Wedged-MIP heuristic running times. For each graph tested other than the 1000- and 2000-vertex graph, we run MCSF and TSD for 24 hours, and if no solution is found we terminate the execution. For the 1000- and 2000-vertex graphs, we allow 168 hours (1 week) before terminating.

Table 7.8: Running times (hh:mm:ss) for Wedged-MIP heuristic, MCSF and TSD

Graph	N	**Wedged-MIP heuristic**	**MCSF**	**TSD**
12-vertex cubic	12	00:00:01	00:00:01	00:00:01
24-vertex cubic	24	00:00:01	00:00:01	00:00:02
38-vertex cubic	38	00:00:01	00:00:01	00:21:04
8×8 Knight's Tour	64	00:00:02	00:00:01	$>$ 24 hours
Perturbed Horton	94	00:00:02	00:03:04	$>$ 24 hours
12×12 Knight's Tour	144	00:00:03	00:01:12	$>$ 24 hours
250-vertex	250	00:00:16	00:29:42	$>$ 24 hours
20×20 Knight's Tour	400	00:20:57	17:35:57	$>$ 24 hours
500-vertex	500	00:10:01	$>$ 24 hours	$>$ 24 hours
1000-vertex	1000	00:30:46	$>$ 1 week	$>$ 1 week
2000-vertex	2000	10:24:05	$>$ 1 week	$>$ 1 week

In summary, in this chapter, we considered the space $\mathcal{X}_\beta$ of discounted occupational measures associated with the embedding of a given graph in a discounted MDP as described in Chapter 4. We demonstrated that when modified by a small number of additional constraints, the resulting space forms a suitable domain for a number of promising algorithmic approaches to the HCP. We feel that this way of tackling the HCP, numerically, can be explored much further. In particular, for graphs with large number of vertices, using a value of β that is very close to 1 could present accuracy problems in verifying validity of linear constraints involving that parameter. It is possible that exploiting techniques similar to those used to develop the parameter-free model of Section 4.5 may help resolve this problem.