Multi-objective discounted dynamic programming

The Neighbour Search approach to construct Pareto sets of multi-objective Markov Decision Processes

Gianluca Dorini · Dragan Savić

Received: October 16, 2009/ Accepted: date

Abstract The Neighbour Search (NS) algorithm, is an iterative method for constructing Pareto sets of multi-dimensional polytopes. A NS iteration consists in two steps: Edges Exploration and Neighbour Detection. Edges Exploration takes a Pareto vertex and determines all Pareto edges connecting such a Pareto vertex to its neighbours. Each neighbour is again a Pareto vertex that is obtained by Neighbour Detection. The procedure continues until all Pareto vertices are explored. The purpose of this paper is to describe in detail the application of NS to Markov Decision Processes (MDPs) with $N$ discounted objectives. Novel numeric techniques are herein developed to effectively adapt Edges Exploration and Neighbour Detection to the MDPs characteristics. Edges Exploration consists of solving a problem of redundancy removal for systems of linear inequalities in $N$ dimensions; the number of inequalities is equivalent to the size of the MDP. Neighbour Detection is performed either by Direct Neighbour Search (DNS) or by Cross Neighbour Search (CNS). The former requires the Bellman equation to be solved, even though with a reduced action set. The latter does not require the Bellman equation to be solved, and is computationally linear in the size of the MDP, and thus more efficient than DNS. However, CNS requires conditions that are not always fulfilled, whereas DNS is always applicable. Experimental results suggest that conditions for CNS to be applicable are actually satisfied for the most of NS iterations. In
particular, CNS feasibility seems to grow with the size of the MDP and with the number of objectives $N$.

**Keywords** Multi-objective optimization · Markov Decision Process (MDP) · Markov Chains · Dynamic Programming

### 1 Introduction

In real life, decision making problems are made in many stages and with many objectives in mind. Problems of this nature require decisions to be taken sequentially over long periods, and their consequences have immediate impacts (measured by rewards, damages, losses, improvements,...), and they also influence all future decisions, and hence, all the future impacts. Thus, it can happen that actions taken today may generate an immediate desirable effect (i.e. reward), but also compromise the future at the same time. A key aspect behind decision support is the evaluation of a solution, which means simulating the effects that such a solution would have on the reality if applied, and then calculating performance measures. However, such an assessment is often hard to carry out, due to the inherently stochastic nature of many problems. In this case, uncertainty must be captured or taken into account, and the decision process is hence formalised as a controlled stochastic system, whose structure is often the structure of a Multi-Objective Markov Decision Process (MDP) model. Such models provide several tools for generating solutions (called policies), for many business and engineering problems. Let $\pi$ be a possible policy, and $V(\pi) \in \mathbb{R}^N$ be the corresponding loss; the goal is to solve the minimization problem

$$V_1(\pi), \ldots, V_N(\pi) \rightarrow \min_{\pi}$$

(1)

with conflicting objectives: the improvement of one criterion leads to the deterioration of other one(s). For this reason, in this context, the types of policies that are investigated are not optimal but Pareto optimal. A policy $\pi$ is Pareto optimal (or simply Pareto), if and only if no other policy $\pi'$ leads to a vector $V(\pi')$ such that $V(\pi) \leq V(\pi')$ (component-wise) and $V(\pi) \neq V(\pi')$. The collection of performances of all Pareto policies is called Pareto set. Decision making can be effectively supported by analysing the tradeoffs among the $N$ objectives; and this can be done once the Pareto set is constructed.

In the literature, several types of multi-objective MDPs have been studied, and the structure of the corresponding Pareto sets have been analyzed. Furukawa [1980] characterized Pareto optimal policies among the non-randomized stationary policies. Henig [1983] confirmed Furukawa’s results, after he studied Pareto optimality among the family of past-dependent, non-randomized policies. Feinberg and Shwartz [1996] characterized the Pareto optimal policies among all policies, establishing that any Pareto optimal point can be attained by two types of policies called respectively $M$-randomized stationary policy and strong $(M, N)$-policy. The paper by Dorini et al. [2007] illustrates a theoretical study of cone-generated orders and corresponding non-dominated so-
olutions, resulting in original statements about topological properties of Pareto sets.

The most common approach adopted in the literature, to addressing multi-objective MDPs is to *sample* the Pareto set, by solving parametric problems, for different sets of parameters. The *scalar approach* reduces the multi-objective problem 1 into a scalar problem

\[ V_1(\pi) \cdot w_1 + \ldots + V_N(\pi) \cdot w_N \rightarrow \min_{\pi} \]  

(2)

where \( w \) is a vector with strictly positive components. Solving problem 2 yields policies that are Pareto optimal for the multi-objective problem 1. By repeatedly solving problem 2 for different vectors \( w \), different Pareto policies are obtained, and their performances are *samples* of the Pareto set. This way to explore the Pareto set is known as the *weighted sum approach*. A detailed illustration of the weighted sum approach to generic multi-objective problems can be found in Johannes [1999]. Pareto policies can be also found by solving *constrained problems* (see, for instance Piunovskiy [1997]), p.100-103), where one criterion is minimized, while \( N - 1 \) constraints are imposed on the other criteria:

\[ V_i(\pi) \leq d_i, i = 1, \ldots, N - 1 \quad V_N(\pi) \rightarrow \min_{\pi} \]  

(3)

By varying the scalars \( d_1, \ldots, d_{N-1} \) within a feasible range, the whole Pareto set can be explored. For solving problems 2 and 3, there are three main methods (Kallenberg [1994]), namely *value iteration*, *policy iteration* and *linear programming*. Value iteration and policy iteration, represent the *dynamic programming approach*, and they are mainly applied for solving the scalar problem 2. Exceptions are papers by Piunovskiy and Mao [2000], Chen and Blankenship [2004], where a dynamic programming approach for solving the constrained problem 3 is described. Linear programming can deal with both problems 2, 3, and with many other non-standard problems involving MDPs. An extensive survey on linear programming applied to MDPs, can be found in Kallenberg [1994]. Other publications showing this wide applicability of linear programming to MDPs, are: Altman [1999], Heyman and Sobel [1994], Piunovskiy [1997]. Hernandez-Lerma and Romera [2004] proved that problem 2 is equivalent to solving the dual of a certain multi-objective linear program. This way of reasoning, called also ‘Convex Analytic Approach’, proved to be effective for solving constrained problems 3. It is very important to emphasize that solving parametric problems and thus *sampling* the Pareto set, can be computationally very hard. In particular, Littman et al. [1995] state that 1) none of the dynamic programming algorithms are known to run in polynomial time in the worst case, and 2) even though linear programming can be solved in polynomial time, the order of the polynomials is large enough that the theoretically efficient algorithms are not efficient in practice.

An alternative to the sampling approach is the approximation approach. For \( \epsilon > 0 \), the \( \epsilon \)-approximate Pareto set contains performances which are not dominated by any other performance of the Pareto set, by a factor of \( 1 + \epsilon \) (Papadimitriou and Yannakakis [2000]). Chatterjee et al. [2006] considered
MDPs with multiple discounted reward objectives, showing that a polynomial-sized \( \epsilon \)-approximate Pareto curve always exists, and it can be computed in polynomial time in the size of the MDP.

This paper describes a methodology to explore the Pareto set of finite discounted MDPs. The methodology is based on the findings reported in the paper Dorini et al. [2007], describing the Neighbourhood Search algorithm, a universal method for constructing Pareto sets for polytopes, when the order is cone-generated. This paper proposes an alternative search methodology, called the Neighbour Search algorithm. The scope of this paper is restricted to standard cone-generated order (i.e. standard multi-objective problems). This paper is divided in two parts. In the first part we consider a generic multi-objective problem, where a Decision Maker has a finite set of solutions and each solution is evaluated with respect of \( N \) performance criteria. Solutions can be randomised, and the corresponding performance set is a convex polytope. The NS algorithm to solve this generic problem is described in section 2. In the second part of this paper we consider the specific problem where the possible solutions are the stationary non-randomised policies to manage an MDP with \( N \) discounted objectives. Randomised policies can be obtained as mixture of non-randomised policies, therefore the corresponding performance set is again a convex polytope. The NS algorithm to solve multi-objective MDPs is described in section 3. The last section summarises the results presented, and the detailed proofs are given in the Appendix.

1.1 Notation and basic definitions

In what follows, the notation adopted throughout the manuscript is introduced. Upper indices \( x^1, x^2, \ldots \) correspond to different vectors in \( \mathbb{R}^N \), whilst lower indices are the vector components: \( x = (x_1, x_2, \ldots, x_N) \in \mathbb{R}^N \). The scalar product \( \sum_{k=1}^{N} x_k y_k \) is denoted with \( \langle x, y \rangle \); ‘iff’ is the abbreviation for ‘if and only if’. If \( A \) is a discrete set, \( |A| \) denotes the number of its elements. If \( B \) is a subset of \( \mathbb{R}^N \), \( \text{ri}(B) \) denotes its relative interior. We denote with \( \text{LP}(d, n) \) the computational complexity of solving a Linear Programming problem with \( d \) decision variables and \( n \) constraints.

\textbf{Definition 1} A set \( D \in \mathbb{R}^N \) is a convex set, iff for every pair of points \( x, y \in D \), the whole segment \( \theta x + (1 - \theta) y, \theta \in [0, 1] \) belongs to \( D \).

\textbf{Definition 2} Given \( K < \infty \) vectors \( b^i \in \mathbb{R}^{N+1}, i = 1, \ldots, K \), a polyhedron is a space \( D \subset \mathbb{R}^N \), resulting from the intersection of halfspaces and hyperplanes in \( \mathbb{R}^N \):

\[ D = \bigcap_{i=1}^{K} \left\{ x \in \mathbb{R}^N \left| \sum_{j=1}^{N} x_j b_{ij} \leq b_{i,N+1} \right\} \right. \]  

Conditions of inequality (case \( \sum_{k=1}^{N} x_k b_k \leq b_{N+1} \) and \( \sum_{k=1}^{N} x_k b_k < b_{N+1} \)) define halfspaces; conditions of strict equality (\( \sum_{k=1}^{N} x_k b_k = b_{N+1} \)) define hyperplanes. A point \( x \in D \) belongs to \( \text{ri}(D) \) iff all conditions of inequality
of formula 4 are fulfilled by \( x \) with \textit{strict} inequality. Bounded polyhedra are called \textit{(convex) polytopes}. An 
affine set in \( \mathbb{R}^N \) is the set of the form \( x + L \), where \( L \subseteq \mathbb{R}^N \) is a linear subspace. The \textit{affine hull} of \( D \subseteq \mathbb{R}^N \), \( \text{aff}(D) \), is the intersection of all affine sets containing \( D \). The affine hull \( \text{aff}(D) \) of any set \( D \subseteq \mathbb{R}^N \) is an affine set: \( \text{aff}(D) = x + L \). The dimension of a set \( D \subseteq \mathbb{R}^N \) is the dimension of \( \text{aff}(D) \), i.e. the dimension of \( L \). For the details see Bertsekas [2003].

A \( d \)-polyhedron/d-polytope, is a polyhedron/polytope with dimension equal to \( d \). A 3-polytope is also simply called a \textit{polytope}, 2-polytopes are also called \textit{polygons}, 1-polytopes are also called \textit{segments}.

**Definition 3** The convex hull of a set \( S \subseteq \mathbb{R}^N \), denoted with \( \text{conv}(S) \), is the intersection of all convex sets that contain \( S \). In the case of finite \( S = \{s^1, \ldots, s^K\} \), the corresponding convex hull is:

\[
\text{conv}(S) = \left\{ \sum_{k=1}^{K} p^k s^k \mid p^k \geq 0, \sum_{k=1}^{K} p^k = 1 \right\}
\]

(5)

The convex hull of a finite set of points is a convex polytope; conversely, a convex polytope is the convex hull of a set of points ([McMullen and Sherphard, 1971, pp. 43-47]). Representations of polytopes in the form 4 is called \( \mathcal{H} \)-representation; whereas form 5 is called \( \mathcal{V} \)-representation.

**Definition 4** Suppose \( D \) is a polytope. A convex subset \( F \) of \( D \) is called \textit{exposed} if representations \( \theta x + (1 - \theta) y \in F, \theta \in [0, 1] \) are possible only if \( x, y \in F \). A hyperplane \( H \subseteq \mathbb{R}^N \) is said to support a set \( D \subseteq \mathbb{R}^N \), and the intersection \( F = D \cap H \) is called a \textit{face} of \( D \), iff \( F \) is an extreme subset of \( D \) ([Dattorro 2005, p. 110]).

Any face of a polytope is again a polytope ([Bertsekas [2003][p. 216]), and any face of it is also a face of the original polytope ([Dattorro, 2005, p. 76]); a non-empty intersection of two faces is again a face ([Stoer and Witzgall, 1970, Proposition 3.6.2]. A \( d \)-polytope is actually a \( d \)-face; \( (d-1) \)-faces are called \textit{facets}; 1–faces are called \textit{edges} and 0–faces are \textit{vertices}. The collection of vertices of a polytope \( D \) is denoted with \( \text{Ver}(D) \). The faces \( F^2, F^3, \ldots \) of a polytope \( D \) can be (partially) ordered by the inclusion. A facet is also said to be a \textit{maximal face} as it is not a strict subset of any face.

We consider the problem to remove redundant inequalities from a system of linear inequalities; namely to remove all redundant halfspaces and hyperplanes of formula 4 defining \( D \subseteq \mathbb{R}^N \) in \( \mathcal{H} \)-representation (see definition 2). Let \( K \) be the number of facets of a \( d \)-polytope \( D \subset \mathbb{R}^N \), we use the notation

\[
[H^1, H^2, \ldots, H^K] := \text{facets}(D)
\]

to refer to an algorithm computing the \( K \) hyperplanes \( H^1, H^2, \ldots, H^K \) supporting \( D \) on its facets \( F^1, F^2, \ldots, F^K \).
Example 1 Consider the following system of linear inequalities in $\mathbb{R}^2$: $-x \leq 0; -y \leq 0; -x + y < 1; -2x + 2y < 2; x + y \leq 3; x \leq 3$. Conditions $-2x + 2y < 2$ and $x \leq 3$ are redundant and can be disregarded. Let $a = (0,0), b = (3,0), c = (1,2), d = (0,2)$; the resulting polygone $D \subset \mathbb{R}^2$ has $K = 3$ facets (edges), i.e. the segments $\overline{ab}, \overline{cd}$ and $\overline{da}$. The facets are supported by the straight lines returned by algorithm $[x = 0, y = 0, x + y = 3] := \text{facets}(D)$.

Line $x = 0$ supports $D$ on facet $\overline{da}$; line $y = 0$ on $\overline{ab}$; line $x + y = 3$ on $\overline{bc}$. Note that, due to the strict inequality $x + y < 1$, the segment $\overline{cd}$ is not a facet of $D$.

The problem of redundancy removal for systems of $n$ linear inequalities in $d$ dimensions is computationally equivalent to solve $K$ Linear Programming problems of size $d \times n$, where $K$ is the number of nonredundant inequalities, namely $K \cdot \text{LP}(d, n)$, Ottmann et al. [1995]. Numeric techniques to solve redundancy removal problems have been widely investigated in the field of computational geometry. For this reason and for the sake of brevity, none of those techniques will be described in this paper. The reader is referred to the relevant literature, for instance Fukuda [1998].

2 The Neighbour Search approach for constructing Pareto sets of convex polytopes

2.1 The problem

We consider a problem where a Decision Maker has a finite set $M$ of possible decisions, and each decision $\varphi \in M$ is associated with a vector of costs.

$$V(\varphi) = (s_1, s_2, \ldots, s_N) \in \mathbb{R}^N$$

Those costs are not necessarily related to money, they quantify generic losses, and thus something to minimize. Different solutions lead to different performances; the collection of all performances of $M$ is a (finite) set $S$ of points of $\mathbb{R}^N$. Solutions can also be randomized: let $p_\varphi$ be the probability for the solution $\varphi \in M$ to be taken, the performance of such a randomized solution is the expectation $\sum_{\varphi \in M} V(\varphi)p_\varphi \in D$, where the performance set $D$ is a convex polytope of $\mathbb{R}^N$, generated as convex hull $D = \text{conv}(S)$. In what follows, we call model a set of decisions $M$, and submodel a subset of a Model. Set $D$ is the performance set, whereas $S$ is the non-randomised performance set, and we say that $S$ and $D$ are generated by the model $M$. In order to compare different solutions, points in $D$ are partially ordered with respect to dominance. A point $x \in \mathbb{R}^N$ is dominated by a point $y \in \mathbb{R}^N$, iff vector $x - y$ has non negative components, and $x \neq y$. A solution whose performance is $x \in D$ is said to be dominated by a solution whose performance is $y \in D$ if $y$ dominates $x$. Obviously, there is no reason for a Decision Maker to prefer $x$ to $y$; in general, there is no reason to consider any dominated solution.
Definition 5 A point \( x \in D \) is called Pareto optimal (or efficient or minimal), if it is not dominated by any point in \( D \). The collection of all Pareto points of a convex set \( D \), called Pareto set, is denoted as \( \text{Par}(D) \).

The Pareto set is a collection of faces of \( D \), that are entirely made of Pareto points. In this paper those faces are referred to as Pareto faces, Pareto \( k \)-faces, Pareto facets, Pareto edges and Pareto vertices. Solutions whose performance belongs to \( \text{Par}(D) \), are called Pareto optimal solutions or simply Pareto solutions. A submodel \( M' \subset M \) generating a Pareto face \( F' = \text{conv}(S') \) is called Pareto submodel. Only Pareto optimal solutions are of interest to the DM.

The methodology herein presented is an algorithm called Neighbour Search (NS), to construct the whole Pareto set of a convex polytope, by means of collection of all Pareto vertices and Pareto edges, and to obtain the corresponding Pareto submodels. The Neighbour Search exploits the property of \( \text{Par}(D) \cap \text{Ver}(D) \) to be a connected graph. Namely for any two points \( s^A, s^B \in \text{Par}(D) \cap \text{Ver}(D) \) there is always a finite sequence \( s^A = s^1, s^2, \ldots, s^L = s^B \), such that \( s^i \) and \( s^{i+1} \) are neighbours, for every \( i = 1, \ldots, L - 1 \) (Dorini et al. [2007]). The name of the algorithm derives from the following definition:

Definition 6 Consider a Pareto Vertex \( s^* \in \text{Par}(D) \cap \text{Ver}(D) \); a point \( s^v \in S \setminus \{s^*\} \) is a neighbour of \( s^* \), iff \( s^v \) is a Pareto vertex of \( D \), and the segment \( F = \text{conv}\{s^*, s^v\} \) is a Pareto edge of \( D \).

The idea of Neighbour Search is this: assumed that it is possible to find all neighbours of a given Pareto vertex, and then the same principle can be recursively applied to explore the whole Pareto set. Based on definition 6, it follows that for a given Pareto vertex \( s^* \in \text{Par}(D) \cap \text{Ver}(D) \) there are as many neighbours of \( s^* \) as the Pareto edges of \( \text{Par}(D) \) containing \( s^* \). We break down the operation to find all neighbours of \( s^* \) in two steps: Edges Exploration and Neighbour Detection. The first step determines all Pareto edges of \( \text{Par}(D) \) containing \( s^* \). The second step obtains from each vertex the included neighbour of \( s^* \).

In this paper we denote with \( M(s^*) \subset M \) the submodel generating a Pareto vertex \( s^* \in \text{Par}(D) \cap \text{Ver}(D) \). Denotation \( M^e(s^*) \subset M \) refers to a submodel whose performance set is a Pareto edge \( F = \text{conv}(S^e(s^*)) \) including \( s^* \); set \( S^e(s^*) \) is the non-randomized performance set of \( M^e(s^*) \).

2.2 The algorithm

The NS algorithm takes as input a model \( M \) generating the non-randomised performance set \( S \subset \mathbb{R}^N \) and the performance set \( D = \text{conv}(S) \); the outputs are two. The first output is a graph \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \) where the nodes \( \mathcal{V} \subseteq S \) are the Pareto vertices \( \mathcal{V} = \text{Par}(D) \cap \text{Ver}(D) \), and the lines \( \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \) are all pairs of Pareto vertices that are neighbours. The second output are the Pareto submodels generating the graph \( \mathcal{G} \). For every vertex \( s^* \in \mathcal{V} \), NS returns a submodel \( M(s^*) \subseteq M \) of all solutions attaining \( s^* \) as performance, i.e. \( V(\varphi) = s^*, \forall \varphi \in M(s^*) \). Each pair of neighbours \( (s^u, s^v) \in \mathcal{V} \) are connected
by a Pareto edge \( \text{conv}(\{s^u, s^v\}) \) which is generated by the submodel \( M(s^u) \cup M(s^v) \). The aim of this section is to describe NS as a procedure combining \textit{Edges Exploration} and \textit{Neighbour Detection} to explore the Pareto set. We first describe the procedure through the example of Figure 1, and then we formalise the algorithm. From now on, unless ambiguous, we will often omit the specifier "Pareto" for faces, vertices, edges, models and submodels.

![Diagram of Neighbour Search](image)

**Fig. 1** Example of Neighbour Search exploration. Circles and triangles represent submodels generating Pareto vertices; \( \bullet \) = passed as input, \( \circ \) = determined using DNS, \( \triangle \) = determined using CNS. Ellipsis represent submodels generating Pareto edges. Solid lines represent neighbour relationships.

Neighbour Search is a recursive procedure where four sets are iterated: \( V, E, M, M^c \). The first two sets define the graph \( G = (V, E) \) of Pareto vertices and edges, as already described in section 2.1. Sets \( M \) and \( M^c \) contain models. Each element of \( M \) is a pair \((s^v, M^v)\) such that \( s^v \) is a vertex and \( M^v = M(s^v) \). Each element of \( M^c \) is a pair \((s^v, M^c)\) such that \( s^v \) is a vertex and model \( M^c \subset M \) generates an edge containing \( s^v \). The NS exploration starts from an initial vertex Pareto vertex \( s^1 \in \text{Par}(D) \cap \text{Ver}(D) \) and the generating model \( M(s^1) \).

\[
V := \{s^1\}, E := \emptyset, M := \{(s^1, M(s^1))\}, M^c := \emptyset
\]

In the diagram of Figure 1A, \( M(s^1) \) is represented as a black circle. For the time being we do not discuss how \( s^1 \) and \( M(s^1) \) were obtained; this will be clarified in section 2.3. Suppose that there are \( K \) edges \( F^1, \ldots, F^K \subset \text{Par}(D) \) containing \( s^1 \); we denote with \( M^c(s^1) \subset M(s^1) \) the submodel generating the \( i \)-th edge \( F^i \). In the example of Figure 1B there are \( K = 4 \) edges, that are represented by oval shapes. Each oval contains the black circle, denoting the
facts that: the edge contains the vertex, and submodel generating the edge contains the submodel generating the vertex. We propose an algorithm to perform Edges Exploration, called edges, returning the models \( M_i^e(s^1) \subset M, i = 1, \ldots, K \) as follows:

\[
\left[ M_1^e(s^1), \ldots, M_K^e(s^1) \right] := \text{edges} \left( M(s^1), M, V(\cdot) \right)
\]

In this section edges is described only in terms of input-output; the description is postponed to section 2.3. Once edges is run, the submodels are then stored in memory:

\[
\mathcal{M}^e := \mathcal{M}^e \cup \{(s^1, M_1^e(s^1)), \ldots, (s^1, M_K^e(s^1))\}
\]

We consider the pair \((s^1, M_i^e(s^1)) \in \mathcal{M}^e\) (any other would have been the same though); the model \( M_i^e(s^1) \) generates the 4-th Pareto edge \( F_4 = \text{conv}(S_i^e(s^1)) \) linking \( s^1 \) to a neighbour \( s^2 \in S_i^e(s^1) \) that we do not know. The goal of Neighbour Detection is to determine \( s^2 \) and \( M(s^2) \). In general, a Pareto edge is the convex hull of a set of points \( s^1, \bar{s}^1, \bar{s}^2, \ldots, s^2 \in \mathbb{R}^N \) that are alligned on a straight line: \( F_4 = \text{conv}(S_i^e(s^1)) = \text{conv}(\{s^1, \bar{s}^1, \bar{s}^2, \ldots, s^2\}) = \text{conv}(\{s^1, s^2\}) \) and \( M_4^e(s^1) = M(s^1) \cup M(\bar{s}^2) \cup M(s^2) \cup \ldots \cup M(s^2) \). The neighbour \( s^2 \) is the point of \( S_i^e(s^1) \) with maximum distance \( \lambda > 0 \) from \( s^1 \). Such a distance can be maximised by taking any point \( \bar{s} \in S_i^e(s^1) \setminus \{s^1\} \) and solving the maximization problem

\[
s^2 := \arg\max_{s \in S_i^e(s^1)} \{\lambda > 0 \mid s = s^1 + \lambda (\bar{s} - s^1)\}
\]

equivalent to the problem \( s^2 := \arg\min_{s \in S_i^e(s^1)} \langle s, \alpha \rangle \) where \( \alpha \in \mathbb{R}^N \) can be either \( s^1 - \bar{s} \) or any other vector with same direction and orientation. The model \( M(s^2) \) is derived as \( M(s^2) := \{\varphi \in M_4^e(s^1) \mid V(\varphi) = s^2\} \). We call this procedure Direct Neighbour Search (DNS). The results are stored in memory:

\[
V := V \cup \{s^2\}, \mathcal{M} := \mathcal{M} \cup \{s^2, M(s^2)\}
\]

Even though we have just found out that \( s^1 \) and \( s^2 \) are neighbours, we do not register this fact by updating \( \mathcal{E} := \mathcal{E} \cup \{s^2\} \); we will do it later. Instead we perform the Edges Exploration on \( s^2 \) and we find \( K = 3 \) models (Figure 1D):

\[
\left[ M_1^e(s^2), M_2^e(s^2), M_3^e(s^2) \right] := \text{edges} \left( M(s^2), M, V(\cdot) \right)
\]

Obviously, among the Pareto edges containing \( s^2 \), there is also the edge \( F_4 \) that we have already explored. In fact the submodel \( M_5^e(s^2) \) coincides with the previously determined submodel \( M_2^e(s^1) \) (Figure 1D). The submodel \( M_2^e(s^2) = M_i^e(s^1) \) cannot be obtained anymore, as it can result from the Edge Exploration on vertex \( s^1 \) or \( s^2 \), and both those vertices have been visited already. At this point we remove the submodel \( M_2^e(s^1) \) from the memory \( \mathcal{M}^e := \mathcal{M}^e \setminus \{(s^1, M_2^e(s^1))\} \), and we register that \( s^1 \) and \( s^2 \) are neighbours \( \mathcal{E} := \mathcal{E} \cup \{(s^1, s^2)\} \) (in Figure 1E this is denoted by replacing the oval with a solid line). We do not store \( M_5^e(s^2) \) in memory, whereas we do store \( M_i^e(s^2) \) and
Step 0

Step 1 IF $\varepsilon$. Step 2 Cross Neighbour Search

Input: An algorithm, an alternative way of DNS to perform Neighbour Detection. In Figure 1H, we observe that models $M_1^0(s^1)$ and $M_2^0(s^1)$ have non empty intersection (similarly $M_1^0(s^1)$ and $M_2^0(s^2)$ have non empty intersection). Since they also do not coincide, as they both belong to $M^*$, it means that they generate two distinct and intersecting Pareto edges: the point of intersection is a new vertex. Such a vertex, marked as $s^4$, is straightforward to obtain, as generated by the model given by the intersection $M(s^4) = M_1^0(s^3) \cap M_2^0(s^1)$, hence $s^4 = V(\varphi), \varphi \in M(s^1)$ (Figure 1H). This procedure is called Cross Neighbour Search (CNS). In section 3.3 we will see in the application to Markov Decision Processes, CNS is preferable to DNS, as computationally more effective. However CNS is applicable only if there are two pairs $(s^u, M^e(s^u)), (s^v, M^e(s^v)) \in M^e$ such that $M^e(s^u) \cup M^e(s^v) \neq \emptyset$. If such a condition is fulfilled we use CNS, otherwise we use DNS.

The Neighbour Search algorithm iterates sets $V, \varepsilon, M$ and $M^e$ by repeatedly performing Edges Exploration and Neighbour Detection. The procedure terminates when $M^e = \emptyset$, hence once all Pareto edges have been explored twice. In the following procedure, whenever a step starts with ‘IF’, a statement follows until ‘THEN’; if the statement is true, one must execute whatever beyond ‘THEN’, otherwise one must execute whatever beyond ‘ELSE’ statement if present, otherwise jump to the next step.

Algorithm 1 Neighbour Search for Convex Polytopes

Input: A model $M$, a function $V : M \rightarrow \mathbb{R}^N$, a Pareto vertex $s^*$ and the submodel $M(s^*) \subset M$.

Step 0 (Initialization). $V := \{s^*\}, \varepsilon := \emptyset, M := \{(s^*, M(s^*))\}, M^e := \emptyset$. Go to Step 4.

Step 1 IF $\varepsilon = \emptyset$ THEN terminate.

Step 2 Cross Neighbour Search.

IF there are two pairs $(s^u, M^e(s^u)), (s^v, M^e(s^u)) \in M^e$ such that

$$M^e(s^u) \cap M^e(s^v) \neq \emptyset$$
THEN determine the Pareto vertex $s^*$, neighbour of both $s^u$ and $s^v$:

$$M(s^*) := M^e(s^u) \cap M^e(s^v)$$

$$s^* := V(\varphi) \subseteq M(s^*)$$

Go to Step 4.

Step 3 Direct Neighbour Search. Select an arbitrary pair $(s^u, M^e(s^u)) \in \mathcal{M}^e$ determine the Pareto vertex $s^*$, neighbour of $s^u$:

$$s^* := \arg\min_{s^u \in S^e(s^u) \setminus \{s^v\}} \langle s, s^u - \tilde{s} \rangle \subseteq \{ \varphi \in M^e(s^u) | V(\varphi) = s^* \}$$

Step 4 Let $K$ be the number of neighbour of $s^*$, determine the submodels $M^e_i(s^*)$, $i = 1, \ldots, K$ using function:

$$[M^e_1(s^*), \ldots, M^e_K(s^*)] := \text{edges}(M(s^*), M, V(\cdot))$$

and set a counter $k := 1$.

Step 5 IF $k = K + 1$ THEN update $\mathcal{V} := \mathcal{V} \cup \{s^*\}$, $\mathcal{M} := \mathcal{M} \cup \{M(s^*)\}$ and to Step 1.

Step 6 IF $\exists (s^u, M^e(s^u)) \in \mathcal{M}^e | M^e(s^u) = M^e_i(s^*)$ THEN update $\mathcal{V} := \mathcal{V} \cap \{s^u, s^*\}$, $\mathcal{M}^e := \mathcal{M}^e \setminus \{(s^u, M^e(s^u))\}$. ELSE $\mathcal{M}^e := \mathcal{M}^e \cup \{(s^*, M^e_i(s^*))\}$.

Step 7 update $k := k + 1$ and go to Step 5.

Output: $\hat{\mathcal{G}} = (\mathcal{V}, \mathcal{E}), \mathcal{M}$.

2.3 Edges Exploration

This subsection describes the edges algorithm for Edges exploration, introduced in section 2.2. Let us disregard the model $M$ for a while, and concentrate on the sets $S$ and $D = \text{conv}(S)$ only. What we want to do is to find points in $\text{Par}(D)$. There are several methods to find Pareto points; one is the scalarization approach: take an arbitrary vector $w$ from the set

$$W = \left\{ w \in \mathbb{R}^N \mid \sum_{i=1}^{N} w_i = 1 \right\} \cap \left\{ w \in \mathbb{R}^N \mid w_j > 0 \right\}$$

it is very well known that the scalar optimisation problem

$$\langle x, w \rangle \to \min_{x \in D}$$

yields a Pareto submodel $M^w$ generating a Pareto face $F^w = D \cap H^w$, supported by the hyperplane

$$H^w = \left\{ x \in \mathbb{R}^N \mid \langle x, w \rangle = \min_{d \in D} \langle d, w \rangle = \min_{s \in S} \langle s, w \rangle \right\}$$
face $F^w$ can be constructed as a convex hull $F^w = \text{conv}(S^w)$ of the finite set $S^w = S \cap H^w$. Vector $w$ is said to be a preference vector of $F^w$ as it yields the Pareto face $F^w$. The collection of preference vectors of a Pareto face is called preference set.

**Definition 7** Given a Pareto face $F = \text{conv}(S^*)$, $S^* \subset S$, a vector $w \in W$ is a preference vector for $F$ and for $S^*$ iff

$$\langle w, x^* \rangle = \min_{s \in S} \langle s, w \rangle, \forall x^* \in F$$

(8)

The collection of all preference vectors for $F$ is called preference set, and it is denoted as $W(F)$:

$$W(F) = W(S^*) = \left\{ w \in W \left| \langle x^*, w \rangle = \min_{s \in S} \langle s, w \rangle, \forall x^* \in F \right. \right\}$$

(9)

In case of a singleton $F = \{x^*\}$, the preference set is simply denoted with $W(x^*)$.

It is very well known (see for instance Wakuta [1996], Heyman and Sobel [1994]) that in case of convex polytopes, it is possible to explore the whole $\text{Par}(D)$ just by solving problem 6 for different vectors $w \in W$:

**Proposition 1** (a) If there is a point $x^* \in \text{Par}(D)$, then there is a vector $w \in W$ such that $w \in W(x^*)$. (b) If there is a point $x^* \in D$, and a vector $w \in W$ such that $w \in W(x^*)$, then $x^* \in \text{Par}(D)$

We study the geometric properties of preference sets to define edges algorithm. The preference set $W(F) = W(S^*)$ of a Pareto face $F = \text{conv}(S^*), S^* \subset S$ is a convex polyhedron, as formula 9 can be re-written as intersection of half-spaces and hyperplanes in $\mathbb{R}^N$:

$$W(F) = W(\text{conv}(S^*)) = W(S^*) = \cdots$$

$$\cdots = W \cap \bigcap_{s \in S \setminus \{x^*\}} \{w \in \mathbb{R}^N | \langle s^* - s, w \rangle \leq 0 \}$$

(10)

Furthermore, the reader can easily verify that $W$ is a $(N-1)$-polytope, hence it is bounded, and thus $W(F) \subset W$ is a convex polytope. The following Proposition establishes an important one-to-one relationship between the dimension of a Pareto face and the dimension of the corresponding preference set.

**Proposition 2** Let $F$ be a Pareto face of $D$. For ($0 \leq k < N$),

a: If $F$ is a $k$-face of $D$, then $W(F)$ is a $(N-k-1)$-polytope.

b: If $W(F)$ is a $(N-k-1)$-polytope, then $F$ is a $k$-face of $D$.

Table 1 shows some example of how, according to proposition 2, dimension of Pareto faces are linked to the dimension of their preference sets. For instance, in a case with three objectives, $N = 3$, if $F$ is a Pareto facet ($k = 2$), then $W(F)$ is a point ($N-k-1 = 0$). If $F$ is a Pareto segment ($k = 1$), $W(F)$ is a segment too $N-k-1 = 1$. If $F$ is a Pareto vertex ($k = 0$), $W(F)$ is a planar polygon $N-k-1 = 2$. 

Based on definition 7, it follows that if two Pareto faces $F_1, F_2$ are ordered by inclusion, $F_1 \subset F_2$, then the inverse order holds for their preference sets: $W(F_2) \subset W(F_1)$. Consequently, the preference set $W(s^*)$ of a Pareto vertex $s^* \in \text{Par}(D) \cap \text{Ver}(D)$ includes the preference set $W(F_j) \subseteq W(s^*)$ of any face $F_j$ including $s^*$. Consider the fact that preference sets of Pareto vertices are $(N-1)$-polytopes, i.e. same dimension of $W$, and suppose that we solve problem 6 with a vector $w$ randomly extracted from $W$ with uniform distribution. Vector $w$ is preference vector for a Pareto vertex only, with probability 1. In fact, the case when $w$ is preference vector for a higher order face is when $w$ belongs to the intersection of preference sets of two or more Pareto vertices, and this has zero probability. The first Pareto vertex and model $s^*, M(s^*)$ that are passed as input to the NS algorithm (algorithm 1), can hence be obtained randomly in the way just described.

**Proposition 3** Consider a vertex $s^* \in \text{Par}(D) \cap \text{Ver}(D)$ and a Pareto edge $F = \text{conv}(S^c(s^*))$, containing $s^*$. For every point $s \in S \setminus \{s^*\}$ consider the hyperplane

$$H(s) = \{ w \in \mathbb{R}^N \mid \langle s^* - s, w \rangle = 0 \} \quad (11)$$

The preference set $W(F) = W(S^c(s^*))$ is i) a facet of $W(s^*)$ and ii) it is supported by $H(s)$ according to the condition

$$W(F) = W(s^*) \cap H(s) \text{ iff } s \in S \setminus \{s^*\} \quad (12)$$

**Corollary 1** Consider a vertex $s^* \in \text{Par}(D) \cap \text{Ver}(D)$ and a Pareto edge $F = \text{conv}(S^c(s^*))$, containing $s^*$. Let $H'$ be the hyperplane supporting $W(s^*)$ on the preference set $W(F) = W(s^*) \cap H'$. Then

$$S^c(s^*) = \{s^*\} \cup \{ s \in S \setminus \{s^*\} \mid H(s) = H' \} \quad (13)$$

the proof is straightforward.

The **edges** algorithm exploits equation 13 to perform Edges exploration. Consider a Pareto vertex $s^* \in \text{Par}(D) \cap \text{Ver}(D)$, let $K$ be the number of facets of the preference set $W(s^*)$. Those facets are the preference sets $W(F^1), W(F^2), \ldots, W(F^K)$ of the $K$ Pareto edges $F^1, \ldots, F^K \in \text{Par}(D)$ containing $s^*$. We use the **facets** algorithm (see section 1.1) to find the hyperplanes

$$[H^1, \ldots, H^K] := \text{facets}(W(s^*))$$

<table>
<thead>
<tr>
<th>dimension of $F$</th>
<th>dimension of $W(F)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table 1** Relationship between the order of a Pareto face $F$ and the order of the corresponding preference set $W(F)$, for different numbers $N$ of objectives.
supporting \( W(s^*) \) on \( W(F^1), W(F^2), \ldots, W(F^K) \) and we use equation 13 to derive the submodels generating the Pareto edges:

**Algorithm 2 edges**

**Input:** A model \( M \), a function \( V: M \rightarrow \mathbb{R}^N \), a submodel \( M(s^*) \subseteq M \) generating the Pareto vertex \( s^* = V(\varphi^*) \in \text{Par}(D) \cap \text{Ver}(D), \varphi^* \in M(s^*) \).

**Step 1** Obtain the hyperplanes \( [H^1, H^2, \ldots, H^K] := \text{facets}(W(s*)) \), where the preference set \( W(s^*) \) is the \((N - 1)\)-polytope in \( H \)-representation:

\[
W(s^*) := W \cap \bigcap_{s \in S \setminus \{s^*\}} \{ w \in \mathbb{R}^N \; | \; \langle s^* - s, w \rangle \leq 0 \} \quad (14)
\]

**Step 2** for all \( k = 1, \ldots, K \) obtain the submodel

\[
M_k^r(s^*) := M(s^*) \cup \{ \varphi^* \in M \setminus M(s^*) \; | 
H^k = H(\varphi^*) \}
\]

Where \( H(\varphi^*) = \{ w \in \mathbb{R}^N \; | \; \langle s^* - V(\varphi^*), w \rangle = 0 \} \)

**Output:** \( M_1^r(s^*), \ldots, M_K^r(s^*) \)

The reader can easily verify that the overall complexity of \textit{edges} coincides with the complexity of \textit{facets}, namely the problem of redundancy removal for systems of \(|S| - 1\) linear inequalities in \( N \) dimensions; i.e. \( K \cdot \text{LP}(N, |S|) \).

3 The Neighbour Search approach for constructing Pareto sets of Multi-Objective Markov Decision Processes

3.1 The problem

So far we have considered generic models being finite sets of decisions. We now concentrate on the special case where a model is a Markov Decision Process (MDP) \( M = \{ X, A, A(\cdot), p(\cdot), g(\cdot) \} \), where \( X \) is a finite state space; \( A \) is a finite action space; \( A(x) \subseteq A \) are the sets of available actions at state \( x \in X \); \( p(y|x, a) \) are transition probabilities form \( X \times A \) to \( X \); \( g(x, a) = (g_1, g_2, \ldots, g_N) \) are an \( N \)-dimensional vectors of costs, where \( (x, a) \in X \times A \). A stationary randomized policy, is a transition probability \( \pi (a|x) \) from \( X \) to \( A \) concentrated on the set \( A(x) \). A stationary non-randomized policy \( \varphi \) provides a deterministic action \( \pi (\varphi (x)|x) = 1 \), for each state \( x \in X \). We denote with \( |A(X)| \) the total number of state-action pairs \( |A(X)| = \sum_{x \in X} |A(x)| \), and with \( |A^N| \) the total number of stationary non-randomised policies \( |A^N| = \prod_{x \in X} |A(x)| \).

**Definition 8 MDP submodels**

1. A submodel \( M^i \) of a model \( M \), denoted with \( M^i \subseteq M \), is identical to \( M \) but with a reduced set of available actions: \( A^i(x) \subseteq A(x), x \in X \). If \( M^i \subseteq M \), then \( A^i(x) \subseteq A(x) \) for at least one \( x \in X \).
2 A model $M^{iii} \subset M$ is given by the intersection of submodel $M^{i} \subset M$ with submodel $M^{iii} \subset M$, and this is denoted with $M^{iii} = M^{i} \cap M^{iii}$ if $A^{iii}(x) = A^{i}(x) \cap A^{iii}(x) \neq \emptyset, x \in X$. The intersection between two models $M^{i}$ and $M^{iii}$ is empty, and it is denoted with $M^{i} \cap M^{iii} = \emptyset$, if there is at least one state $x \in X$, such that $A^{i}(x) \cap A^{iii}(x) = \emptyset$.

3 A model $M^{iv}$ is given by the union of $M^{i}$ with $M^{iii}$, iff $A(x) = (A^{i}(x) \cup A^{iii}(x)) \setminus (A^{i}(x) \cap A^{iii}(x))$, for all $x \in X$.

4 The membership of a policy $\pi, \varphi$ to a submodel $M^{i}$ is denoted with $\pi \in M^{i}$, $\varphi \in M^{i}$.

According to Ionescu Tulcea theorem [Bertsekas and Shreve, 1978, Dynkin and Yushkevich, 1979], [Piunovskiy, 1997, Theorem A1.11], a policy $\pi$ and an initial probability distribution $\mu$ on $X$ define a unique probability distribution $\mathbb{P}_{\mu}$ on the space of trajectories $(X \times A)^{\infty} = (x_{0}, a_{0}, x_{1}, \ldots)$. The corresponding mathematical expectation is denoted by $\mathbb{E}_{\mu}$. For a fixed initial distribution $\mu$ the performance of a policy $\pi$ is evaluated by a vector $V(\pi) = (V_{1}(\pi), V_{2}(\pi), \ldots, V_{N}(\pi))$, representing the Total Discounted Cost (TDC), given a discount factor $\beta \in (0, 1)$:

$$V_{i}(\pi) = \mathbb{E}_{\mu}^{\pi} \left[ \sum_{t=0}^{\infty} \beta^{t} g_{i}(x_{t}, a_{t}) \right]$$

The performance set $D$ is the set of all possible vectors $V(\pi)$ under different policies $\pi$. The non-randomized performance set $S$ is the set of all possible vectors $V(\varphi)$ under different non randomised policies $\varphi$ (it is a finite set of points, as the number of total non-randomized policies is finite). It is well known (see for instance [Dorini et al., 2007, Remark 4]), that $D$ is the polytope resulting from the convex hull $D = \text{conv}(S)$. In particular, stationary non-randomised policies can be combined in many ways (convex combinations), to create other policies (mixtures), whose performances can correspond to any point of $D$. This property also applies to subsets of $D$; consider a hyperplane $H$ supporting $D$ on a face $F = D \cap H$, and $S$ on a subset $S \cap H$; then $F = \text{conv}(H \cap S)$. For further reading on this topic, more details can be found in papers by [Feinberg and Shwartz, 1996, Feinberg, 2000] and in monographs by [Heyman and Sobel, 1994, Piunovskiy, 1997].

**Definition 9** A policy $\pi$ is called Pareto optimal policy or Pareto policy, if $V(\pi) \in \text{Par}(D)$. Some authors call a policy Pareto optimal, only if it gives a Pareto point for any initial state Wakuta [1998].

We develop numeric techniques to effectively adapt NS to the case where the model $M$ is an MDP with initial distribution $\mu(x) > 0, \forall x \in X$. The resulting algorithm, discussed in section 3.4, is almost identical to algorithm 1. The only difference is that Edges Exploration and Neighbour Detection are performed by numeric techniques that have been developed based on the MDP's characteristics. Those techniques are discussed in sections 3.2 and 3.3.
3.2 Edges Exploration for MDPs

Pareto policies can be obtained using the Dynamic Programming approach (DP), based on the following relationship:

\[
\langle V(\pi), w \rangle = E_{\mu}^{\pi} \left[ \sum_{t=0}^{\infty} \beta^t \langle g(x_t, a_t), w \rangle \right]
\]

making problem \( (d, w) \rightarrow \min_{d \in D} d \) equivalent to the problem

\[
\langle V(\pi), w \rangle \rightarrow \min_{\pi}
\]

In order to solve problem 16, one has to determine the Bellman function \( v^w: X \rightarrow \mathbb{R} \), such that

\[
\langle V(\pi), w \rangle = \sum_{x \in X} v^w(x) \mu(x)
\]

is the unique solution of the Bellman equation

\[
v^w(x) = \min_{a \in A(x)} \left[ \langle g(x, a), w \rangle + \beta \sum_{y \in X} p(y|x, a) v^w(y) \right] x \in X
\]

Then, all policies of the submodel \( M^w \subset M \), with the following set of actions

\[
A^w(x) = \arg\min_{a \in A(x)} \left[ \langle g(x, a), w \rangle + \beta \sum_{y \in X} p(y|x, a) v^w(y) \right] x \in X
\]

are Pareto policies.

**Proposition 4** [Pinovskiy, 1997, p.53] Given a vector \( w \in W \), a policy \( \pi^* \) attains the minimum \( \langle V(\pi^*), w \rangle = \min_{\pi} \langle V(\pi), w \rangle \) if and only if \( \pi^*(x) \in A^w(x) \) for every \( x \in X \).

The non-randomised performance set \( S^w \) is the set of all performance vectors generated by all non-randomized policies in the submodel \( M^w \). Similarly, the performance set \( F^w = \text{conv}(S^w) \) is the face of \( D \) coinciding with the total performance set in the submodel \( M^w \).

**Definition 10** A submodel \( \bar{M} \subseteq M \) is called the Bellman submodel if there is a vector \( w \in W \) such that \( \bar{M} \) is generated by \( \bar{M} \).

Bellman equation can be represented as a Linear Programming problem of \(|X|\) decision variables subject to \(|A(X)|\) constraints, and solved in polynomial time. The complexity of solving the Bellman equation is therefore assumed to be \( \text{LP}(|X|, |A(X)|) \). However, the degree of the polynomial is nontrivial and the methods that are guaranteed to achieve such polynomial time performance do not make any significant use of the structure of MDPs (Littman et al., 1995). Bellman equation can also be solved using value iteration and policy iteration, (see Kallenberg [2002], Bertsekas [2000]). In this paper, whenever we refer to MDP model/submodels we always imply Bellman model/submodels.
Consistently with the denotation adopted throughout this manuscript, we denote with \(M(s^*) \subset M\) the (Bellman) submodel generating a Pareto vertex \(s^* \in \text{Par}(D) \cap \text{Ver}(D)\), and with \(A(s^*, x) \subset A(x), x \in X\) the corresponding action set. Denotation \(M^e(s^*) \subset M\) refers to a (Bellman) submodel whose performance set is a Pareto edge \(F = \text{conv}(S^e(s^*))\) including \(s^*\); set \(S^e(s^*)\) is the non-randomized performance set of \(M^e(s^*)\), and \(A^e(s^*, x) \subset A(x), x \in X\) is the action set.

The algorithm \textbf{edges}, described in section 2.3, performs Edges Exploration by computing the facets of the preference set \(W(s^*)\) of a vertex \(s^* \in \text{Par}(D) \cap \text{Ver}(D)\). This consists on using algorithm \textbf{facets} to solve a problem of redundancy removal for linear inequality systems, and hence polytope \(W(s^*)\) must be given in \(H\)-representation, i.e. in the form of formula 14:

\[
W(s^*) = \bigcap_{\nu \in S^1(s^*)} \{w \in W | (s^* - s, w) \leq 0\}. 
\]

This requires one to know the coordinates of all points of \(S\), and hence to evaluate the performance of each possible non-randomized policy. The performance \(s = V(\phi) \in S\) of a stationary non-randomized policy \(\phi \in M\), is determined in two steps. First compute a function \(J : X \to \mathbb{R}^N\), called cost-to-go, in time \(O(N |X|^3)\) by solving \(N\) systems of \(|X|\) linear equations with \(|X|\) unknowns:

\[
J_i(\phi, x) = g_i(x, \phi(x)) + \beta \sum_{y \in X} p(y | x, \phi(x)) J_i(\phi, y) \\
\hspace{1cm} x \in X \quad i = 1, \ldots, N 
\]  

(19)

Secondly, obtain the components of the performance \(V_1(\phi), \ldots, V_N(\phi)\) in time \(O(N |X|)\) as \(V_i(\phi) = \sum_{x \in X} \mu(x) J_i(\phi, x)\). Different methods of policy evaluation can be found in Bertsekas [2000]. In order to construct set \(S\), the set of equations 19 must be solved as many times as the number of the stationary non-randomised policies \(|A^X| = \prod_{x \in X} |A(x)|\), and this takes time \(O(N |X|^3 |A^X|)\). We conclude that constructing \(S\) is unfeasible for many practical cases. Furthermore, suppose that \(S\) is constructed; in the worst case we have as many points in \(S\) as stationary non-randomised policies in \(M\). Since \textbf{edges} takes time \(K \cdot \text{LP}(N, |S|)\) (see section 2.3), hence for the MDP case it takes time \(K \cdot \text{LP}(N, |A^X|)\). Again, even having set \(S\) constructed, running \textbf{edges} would be practically unfeasible. We show how this issue can be effectively overcome by exploiting the realtionships between the Bellman function and cost-to-go.

**Proposition 5** Consider a Pareto face \(F\) generated by a Bellman submodel \(M^* \subset M\). For every stationary non-randomized policy \(\phi^* \in M^*\)

\begin{itemize}
\item[i:] \(v^w(x) = \langle J(\phi^*, x), w \rangle^*\), for every \(x \in X\) if \(w \in W(F)\).
\item[ii:] \(v^w(x) \neq \langle J(\phi^*, x), w \rangle^*\), for at least one state \(x \in X\) if \(w \notin W(F)\).
\end{itemize}

**Proposition 6** Let \(s^* \in \text{Par}(D) \cap \text{Ver}(D)\), all stationary non-randomized policies of \(M(s^*)\) attain the same cost-to-go; i.e. \(J(\phi^1, x) = J(\phi^2, x)\), \(\forall x \in X\) for every \(\phi^1, \phi^2 \in M(s^*)\).

Whenever we use denotation \(J(s^*, \cdot)\) with \(s^*\) being a Pareto vertex, we refer to the cost-to-go function fulfilling equation 19, which is the same for every stationary non-randomized policies of \(M(s^*)\), as stated in proposition 6.
**Definition 11** The Q-function of a Pareto vertex \( s^* \in \text{Par}(D) \cap \text{Ver}(D) \) is the function \( Q : X \times A \rightarrow \mathbb{R}^N \)

\[
Q_i(s^*, x, a) = g_i(x, a) + \beta \sum_{y \in X} p(y|x, a) J_i(s^*, y)
\]

\( x \in X \ a \in A(x) \ i \in 1, \ldots, N \)  \hfill (20)

Obtained from the cost-to-go \( J(s^*, \cdot) \) in time \( O(N|X||A(X)|) \).

**Proposition 7** Consider a Bellman submodel \( M(s^*) \subset M \) generating the Pareto vertex \( s^* \in \text{Par}(D) \cap \text{Ver}(D) \). Take an arbitrary stationary non-randomized policy \( \varphi^* \in M(s^*) \). The preference set \( W(s^*) \) is constructed as follows:

\[
W(s^*) = W \cap \bigcap_{(x,a)} \{ w \in \mathbb{R}^N | Q(s^*, x, \varphi^*(x)) - Q(s^*, x, a), w \leq 0 \}
\]

subject to:

\[
(x, a) \in X \times A \ | x \in X, a \in A(x) \ \text{\setminus} \ A(s^*, x)
\]  \hfill (21)

The Q-function contains the necessary information to represent preference sets of Pareto vertices as a system of linear inequalities, without having to construct \( S \). Furthermore, the maximum number of inequalities of formula 21 is the equal to \( |A(X)| - |X| \); namely the case where \( M(s^*) \) has one policy only. This is a very important result, because even if we assume to have the coordinates of all points of \( S \), in the worst case there would be \( |A^X| - |X| \) inequalities. Based on the concept of proposition 7, we re-formulate proposition 3 and corollary 1 for the MDP-specific case.

**Proposition 8** Consider a Bellman submodel \( M(s^*) \) generating the Pareto vertex \( s^* \in \text{Par}(D) \cap \text{Ver}(D) \). Let \( F = \text{conv}(S^*(s^*)) \) be a Pareto edge containing \( s^* \). Take an arbitrary stationary non randomized policy \( \varphi^* \in M(s^*) \), and associate each pair \( (x, a) \in X \times A \ | x \in X, a \in A(x) \ \text{\setminus} \ A(s^*, x) \) to the hyperplane

\[
H(x, a) = \{ w \in \mathbb{R}^N | Q(s^*, x, \varphi^*(x)) - Q(s^*, x, a), w \leq 0 \}
\]  \hfill (22)

The preference set \( W(F) = W(S^*(s^*)) \) is i) a facet of \( W(s^*) \) and ii) it is supported by \( H(x, a) \) according to the condition

\[
W(F) = W(s^*) \cap H(x, a) \iff x \in X, a \in A^e(s^*, x) \ \text{\setminus} \ A(s^*, x)
\]  \hfill (23)

where \( A^e(s^*, x) \subset A(x), \forall x \in X \) is the action set of the submodel \( M^e(s^*) \) generating \( F \) and \( S^e(s^*) \).

**Corollary 2** Consider a vertex \( s^* \in \text{Par}(D) \cap \text{Ver}(D) \) and a Pareto edge \( F = \text{conv}(S^*(s^*)) \) containing \( s^* \). For every state \( x \in X \), the action set \( A^e(s^*, x) \) of the model \( M^e(s^*) \) generating \( F \) is constructed as follows

\[
A^e(s^*, x) = A(s^*, x) \cup \{ a \in A(x) \ \text{\setminus} \ A(s^*, x) | H(x, a) = H' \}
\]  \hfill (24)

Where \( H' \) is the hyperplane supporting \( W(s^*) \) on the preference set \( W(F) = W(s^*) \cap H' \). The proof is straightforward.
Suppose that a Pareto vertex \( s^* \in \text{Par}(D) \cap \text{Ver}(D) \) belongs to \( K \) edges, the Q-function \( Q(s^*, \cdot, \cdot) \) contains the necessary information to construct all submodels \( M^*_k(s^*) \) generating all edges \( k = 1, \ldots, K \). This is done using the following procedure, which is an adaptation of algorithm \textit{edges} to the MDP case.

**Algorithm 3 \textit{edgesMDP}**

**Input:** An MDP model \( M \), a submodel \( M(s^*) \subset M \) generating the Pareto vertex \( s^* \in \text{Par}(D) \cap \text{Ver}(D) \). The Q-function \( Q(s^*, \cdot, \cdot) \).

**Step 1** Obtain the hyperplanes \( [H^1, H^2, \ldots, H^K] : = \text{facets}(W(s^*)) \), where the preference set \( W(s^*) \) is the \((N - 1)\)-polytope in \( \mathcal{H} \)-representation defined in formula 21; namely

\[
W(s^*) = W \cap \bigcap_{(x,a)} \{ w \in \mathbb{R}^N \mid (Q(s^*, x, \varphi^*(x)) - Q(s^*, x, a), w) \leq 0 \}
\]

subject to:

\[
(x, a) \in X \times A \mid x \in X, a \in A(x) \setminus A(s^*, x)
\]

where \( \varphi^* \) is an arbitrary stationary non-randomized policy of \( M(s^*) \).

**Step 2** Take an arbitrary stationary non-randomized policy \( \varphi^* \in M(s^*) \), and obtain for all \( k = 1, \ldots, K \) the submodel \( M^*_k(s^*) \) whose action set \( A^*_k(x, s^*) \) is defined, for all \( x \in X \), as follows:

\[
A^*_k(s^*, x) : = A(s^*, x) \cup \{ a \in A(x) \setminus A(s^*, x) \mid H(x, a) = H^k \}
\]

where \( H(x, a) = \{ w \in \mathbb{R}^N \mid (Q(s^*, x, \varphi^*(x)) - Q(s^*, x, a), w) = 0 \} \)

**Output:** \( M^*_1(s^*), \ldots, M^*_K(s^*) \)

The reader can easily verify that the overall complexity of \textit{edgesMDP} coincides with the complexity of \textit{facets}, namely the problem of redundancy removal for systems of \( |A(X)| - |A(s^*, X)| \) linear inequalities in \( N \) dimensions; i.e. \( K \cdot \text{LP}(N, |A(X)|) \).

3.3 Neighbour Detection for MDPs

In section 2.2 we described the Direct Neighbour Search (DNS) and the Cross Neighbour Search (CNS) to perform Neighbour Detection for the generic case, where models are sets of decisions. In this section we adapt those techniques to the MDP case. As described at Step 3 of algorithm 1, DNS is a straightforward procedure to obtain the unknown neighbour \( s^n \) of a known vertex \( s^* \), given the model \( M^c(s^*) \), generating the vertex linking \( s^* \) and \( s^n \). However, DNS requires the non-randomized performance set \( S^c(s^*) \) to be known, and for the MDP case this is actually not the case. In fact, the \textit{edgesMDP} algorithm just described only provides \( M^c(s^*) \); so if one wants to construct set \( S^c(s^*) \), the set of equations 19 must be solved as many times as the number of the stationary non-randomised policies in \( M^c(s^*) \). Although for some practical cases, such a number of policies may be computationally affordable, we show that for the MDP case it is not necessary to construct \( S^c(s^*) \).
Proposition 9 Consider a model $M^c(s^*)$ generating a Pareto edge $F$, linking two neighbours Pareto vertices $s^*, s^u$.

\begin{enumerate}
\item For every $x \in X$, the action set $A(s^u, x)$ of the submodel $M(s^u)$ is constructed as
\begin{equation}
A(s^u, x) = \arg \min_{a \in A^c(s^*, x)} \left[ g(x, a), \alpha \right] + \beta \sum_{y \in X} p(y|x, a) \hat{v}^\alpha(y)
\end{equation}
\item For every pair $x, a$ such that $x \in X, a \in A(x)$, the two Q-functions $Q(s^*, x, a)$ and $Q(s^u, x, a)$ are linked by the linear relationship
\begin{equation}
Q(s^u, x, a) = Q(s^*, x, a) + \alpha t(x, a)
\end{equation}
\end{enumerate}

where $v^\alpha(x)$ is the Bellman function solving equation
\begin{equation}
v^\alpha(x) = \min_{a \in A^c(s^*, x)} \left[ g(x, a), \alpha \right] + \beta \sum_{y \in X} p(y|x, a) \hat{v}^\alpha(y)
\end{equation}
and $\alpha \in \mathbb{R}^N$ is a vector
\begin{equation}
\alpha = Q(s^*, \bar{x}, a^i) - Q(s^*, \bar{x}, a^{ii})
\end{equation}
defined by any arbitrary $\bar{x}, a^i, a^{ii}$ such that $\bar{x} \in X, a^i \in A(s^*, \bar{x})$ and $a^{ii} \in A^c(s^*, \bar{x}) \setminus A(s^*, \bar{x})$.

The model $M(s^u)$ is obtained by solving Bellman equation 26 with a reduced action set, i.e. $A^c(s^*, x) \subset A(x)$ instead of the whole action set $A(x), \forall x \in X$. This obviously makes equation 26 computationally less intensive to solve than equation 17, as the complexity of the operation is $\text{LP}(|X|, |A^c(s^*, X)|)$ instead of $\text{LP}(|X|, |A(X)|)$. Furthermore, we notice that even for DNS, the Q-function $Q(s^*, \cdot, \cdot)$ is required. The same Q-function was in fact already utilised by algorithm \texttt{edgesMDP} to perform Edges Exploration. For DNS we use $Q(s^*, \cdot, \cdot)$ to define vector $\alpha$ of equation 27. Once equation 26 is solved, we use function $\hat{v}^\alpha$ to construct the action set of submodel $M(s^u)$ (equation 25).

At this point we have to determine the coordinates of the neighbour $s^u$ and we also need to compute the Q-function $Q(s^u, \cdot, \cdot)$, which is necessary to find the neighbours of $s^u$ during the next NS iterations. One way to do this is to compute the cost-to-go in time $O(N|X|^3)$ by solving equations 19, and then obtain the Q-function from the cost-to-go in time $O(N|A(X)||X|)$ using formula 20. Alternatively and even more efficiently, we derive the Q-function directly according to proposition 9. The reader can easily verify that both equations 29 and 28 are computed in time $O(N|A(X)|)$. We formalise the DNS adaptation to the MDP case with the following algorithm.
Algorithm 4 `dnsMDP`

**Input:** An MDP model \( M' \) generating a Pareto vertex \( s^* \in \text{Par}(D) \cap \text{Ver}(D) \). The Q-function \( Q(s^*, \cdot, \cdot) \). A model \( M'' \) generating a Pareto edge \( F \), containing \( s^* \).

**Step 1** Take any arbitrary \( \bar{x}, a^i, a^{ii} \) such that \( \bar{x} \in X \), \( a^i \in A(s^*, \bar{x}) \) and \( a^{ii} \in A''(s^*, \bar{x}) \setminus A(s^*, \bar{x}) \) and define the vector \( \alpha \in \mathbb{R}^N \):

\[
\alpha = Q \left( s^*, \bar{x}, a^i \right) - Q \left( s^*, \bar{x}, a^{ii} \right)
\]

**Step 2** Use \( \alpha \) to scalarise the vectors of costs \( \langle g(\cdot, \cdot), \alpha \rangle \) and compute the function \( \hat{v}^\alpha : X \to \mathbb{R} \) solving the Bellman equation:

\[
\hat{v}^\alpha(x) = \min_{a \in A'(s^*, x)} \left[ \langle g(x, a), \alpha \rangle + \beta \sum_{y \in X} p(y|x, a) \hat{v}^\alpha(y) \right]
\]

for every \( x \in X \).

**Step 3** Construct the function \( \hat{Q}^\alpha : X \times A \to \mathbb{R} \)

\[
\hat{Q}^\alpha(x, a) = \langle g(x, a), \alpha \rangle + \beta \sum_{y \in X} p(y|x, a) \hat{v}^\alpha(y)
\]

for every \( x \in X, a \in A'(s^*, x) \).

**Step 4** The Pareto edge \( F \) links vertex \( s^* \) to a neighbour \( s_u \in \text{Par}(D) \cap \text{Ver}(D) \) generated by model \( M(s_u) \) whose action set is determined as

\[
A(s_u, x) := \arg\min_{a \in A''(s^*, x)} \hat{Q}^\alpha(x, a)
\]

for every \( x \in X \).

**Step 5** Construct the function \( t : X \times A \to \mathbb{R} \)

\[
t(x, a) := \frac{\hat{Q}^\alpha(x, a) - \langle Q(\varphi^*, x, a), \alpha \rangle}{\langle \alpha, \alpha \rangle}
\]

for every \( x \in X, a \in A(x) \).

**Step 6** Construct the Q-function:

\[
Q(s_u, x, a) := Q(s^*, x, a) + \alpha t(x, a)
\]

for every \( x \in X, a \in A(x) \).

**Step 7** Take an arbitrary policy \( \varphi^u \in M(s_u) \) and determine the neighbour

\[
s_u := \sum_{x \in X} J(s_u, x) \mu(x) = \sum_{x \in X} Q(s_u, x, \varphi^u(x)) \mu(x)
\]

**Output:** \( s_u, M(s_u), Q(s_u, \cdot, \cdot) \).
Although the dnsMDP procedure allows for the determination of the Q-function in time $O(N |A(X)|)$, the overall complexity is $\mathbf{LP}(|X|, |A^c(s^*, X)|)$, namely equal to the complexity of solving the Bellman equation 26.

Cross Neighbour Search is applicable whenever there are two distinct submodels $M^c(s^u)$ and $M^c(s^v)$ with non empty intersection $M^c(s^u) \cap M^c(s^v) \neq \emptyset$ (see section 2.2). As explained at Step 2 of algorithm 1, the vertex $s^*$, neighbour of both vertices $s^u$ and $s^v$, is generated by the model resulting from the intersection $M(s^*) := M^c(s^u) \cap M^c(s^v)$. For the MDP case this simply consists in constructing the action set

$$A(s^*, x) := A^c(s^u, x) \cap A^c(s^v, x).$$

for all $x \in X$. We show that proposition 9 can be exploited to efficiently derive the Q-function $Q(s^*, \cdot, \cdot)$ from the Q-functions $Q(s^u, \cdot, \cdot)$ and $Q(s^v, \cdot, \cdot)$. Using formula 27 we define two vectors $\alpha^u, \alpha^v \in \mathbb{R}^N$ such that

$$Q_i (\varphi^u, x, a) = Q_i (\varphi^*, x, a) + \alpha^u_t(x, a)$$
$$Q_i (\varphi^v, x, a) = Q_i (\varphi^*, x, a) + \alpha^v_t(x, a)$$

$i = 1, \ldots, N$ (30)

Hence for every $x \in X, a \in A(x)$ we obtain the Q-function $Q(s^*, \cdot, \cdot)$ by solving a system of $2 \cdot N$ linear equations with $N + 2$ unknowns

$$Q_1(s^*, x, a), \ldots, Q_N(s^*, x, a), t^u(x, a), t^v(x, a)$$

Hence in this case we do not have to solve equations 26 and 29 to obtain $t^u(\cdot, \cdot)$ and/or $t^v(\cdot, \cdot)$. Similarly as for DNS, once we have $Q(s^*, \cdot, \cdot)$ we calculate the coordinates of $s^*$. The CNS algorithm is implemented in the following stepwise procedure, where the system of linear equations 30, overdetermined for $N > 2$, is solved using the least square method.

**Algorithm 5 cnsMDP**

**Input:** Two distinct MDP submodels $M(s^u), M(s^v)$ generating two Pareto vertices $s^u, s^v \in Par(D) \cap Ver(D)$. Two distinct models $M^c(s^u), M^c(s^v)$ generating two Pareto edges, such that $M^c(s^u) \cap M^c(s^v) \neq \emptyset$. The Q-functions $Q(s^u, \cdot, \cdot)$ and $Q(s^v, \cdot, \cdot)$.

**Step 1** For all $x \in X$, construct the action set

$$A(s^*, x) := A^c(s^u, x) \cap A^c(s^v, x).$$

**Step 2** Take any arbitrary $\bar{x}, a^l, a^u$ such that $\bar{x} \in X$, $a^l \in A(s^u, \bar{x})$ and $a^u \in A^c(s^u, \bar{x}) \setminus A(s^u, \bar{x})$ and define the vector $\alpha^u \in \mathbb{R}^N$:

$$\alpha^u = Q(s^u, \bar{x}, a^l) - Q(s^u, \bar{x}, a^u)$$

Similarly, obtain $\alpha^v \in \mathbb{R}^N$ from $Q(s^v, \cdot, \cdot)$.

**Step 3** Calculate the scalar products:

$$a := \langle \alpha^u, \alpha^u \rangle, b := -\langle \alpha^u, \alpha^v \rangle, c := b, d := \langle \alpha^v, \alpha^v \rangle$$
Step 4 For all \( x \in X, a \in A(x) \) compute the Q-function
\[
e := (Q(s^u, x, a) - Q(s^v, x, a), \alpha^u), f := (Q(s^u, x, a) - Q(s^v, x, a), \alpha^v)\]
\[
Q(s^*, x, a) := Q(s^u, x, a) + \alpha^u \frac{de - bf}{ad - be}
\]
Step 7 Take a policy \( \varphi^u \in M(s^*) \) and determine the neighbour
\[
s^* := \sum_{x \in X} J(s^u, x) \mu(x) = \sum_{x \in X} Q(s^u, x, \varphi^u(x)) \mu(x)
\]
Output: \( s^*, M(s^*), Q(s^*, \cdot, \cdot) \).

The reader can easily verify that the overall complexity of cnsMDP is \( O(N \vert A(X) \vert) \) and thus it is more efficient than dnsMDP, taking time LP(\( \vert X \vert, \vert A^p(s^*, X) \vert \)).

3.4 The algorithm

In this section we equip the NS algorithm described in section 2.2, with the MDP-specific methods for Edges Exploration and Neighbour Detection described in sections 3.2 and 3.3. We present a stepwise procedure resulting from the inclusion into algorithm 1, of the edgesMDP, dnsMDP, and cnsMDP algorithms. Since all tree algorithms require the use of Q-functions, and thus Q-functions must be stored in memory, we replace set \( M^c \), with set \( M^c_Q \), whose element is a tuple \( (s, M^c, Q(\cdot, \cdot)) \) such that \( s^v \) is a vertex, model \( M^c \subset M \) generates an edge containing \( s \), and \( Q(\cdot, \cdot) \) is the Q-function of \( s \). Such a Q-function is created when \( s \) is detected, and it stays in memory as long as required by dnsMDP or cnsMDP, in order to detect all neighbours of \( s \).

The first vertex \( s^* \) can be obtained as suggested in section 2.3. Namely solve Bellman equation 17 with a vector \( w \) randomly extracted from \( W \) with uniform distribution and construct model \( M(s^*) \) using formula 18. Then take an arbitrary policy \( \varphi^* \in M(s^*) \), obtain the cost-to-go \( J(s^*, \cdot) \) by solving \( N \) systems of \( \vert X \vert \) linear equations with \( \vert X \vert \) unknowns (equations 19). Determine the vertex coordinates \( s^* := \sum_{x \in X} J(s^*, x) \mu(x) \) and determine the Q-function from the cost-to-go using formula 20.

Algorithm 6 Neighbour Search for MDPs with \( N \) objectives.

Input: A MDP model \( M \), a Pareto vertex \( s^* \), the Bellman submodel \( M(s^*) \subset M \) and the Q-function \( Q(s^*, \cdot, \cdot) \).

Step 0 (Initialization). \( V := \{ s^* \}, \mathcal{E} := \emptyset, \mathcal{M} := \{(s^*, M(s^*))\}, M^c_Q := \emptyset \). Go to Step 4.

Step 1 IF \( \mathcal{E} = \emptyset \) THEN terminate.

Step 2 Cross Neighbour Search.

IF there is a \( (s^u, M^c(s^u), Q(s^u, \cdot, \cdot)) \in M^c_Q \), and a \( (s^v, M^c(s^v), Q(s^v, \cdot, \cdot)) \in M^c_Q \) such that
\[ M^c(s^u) \cap M^c(s^v) \neq \emptyset \]
\[ \begin{align*}
\text{THEN} & \text{ determine the Pareto vertex } s^*, \text{ neighbour of both } s^u \text{ and } s^v: \\
\left[ s^*, M(s^*), Q(s^*, \cdot) \right] & := \text{cnsMDP}(M^u(s^u), Q(s^u, \cdot), M^v(s^v), Q(s^v, \cdot)) \\
\text{Go to Step 4.}
\end{align*} \]

**Step 3** Direct Neighbour Search. Select an arbitrary \((s^u, M^u(s^u), Q(s^u, \cdot)) \in M_Q^u\) determine the Pareto vertex \(s^*\), neighbour of \(s^u\):
\[ \left[ s^*, M(s^*), Q(s^*, \cdot) \right] := \text{dnsMDP}(M(s^u), M^u(s^u), Q(s^u, \cdot)) \]

**Step 4** Let \(K\) be the number of neighbour of \(s^*\), determine the submodels \(M^e_i(s^3), i = 1, \ldots, K\) using function:
\[ \left[ M^e_1(s^*), \ldots, M^e_K(s^*) \right] := \text{edgesMDP}(M, M(s^*), Q(s^*, \cdot)) \]
and set a counter \(k := 1\).

**Step 5** IF \(k = K + 1\)
\[ \text{THEN} \quad \text{update } \mathcal{V} := \mathcal{V} \cup \{s^*\}, \mathcal{M} := \mathcal{M} \cup \{M(s^*)\} \text{ and go to Step 1.} \]

**Step 6** IF \(\exists(s^u, M^u(s^u), Q(s^u, \cdot)) \in M_Q^u \mid M^u(s^u) = M^e_i(s^*)\)
\[ \text{THEN} \quad \text{update } \mathcal{V} := \mathcal{V} \cup \{s^u, s^*\}, \mathcal{M}_Q := \mathcal{M}_Q \setminus \{M(s^u, M^u(s^u), Q(s^u, \cdot))\}. \]
ELSE \( M_Q^e := M_Q^e \cup \{s^u, M^e_i(s^*), Q(s^*, \cdot)\} \).

**Step 7** update \(k := k + 1\) and go to Step 5.

**Output:** \( \mathcal{G} = (\mathcal{V}, \mathcal{E}), \mathcal{M}. \)

---

**Fig. 2** NS exploration of a MDP with \(N = 3, |X| = 5, |A| = 6\). Left: space \(W\) divided in Pareto Vertices preference sets. Right: Pareto Vertices and Pareto edges.

Neighbour Search performs Neighbour Detection in time \(O(N |A(X)|)\) whenever CNS applicable, and in time \(LP(|X|, |A(s^*, X)|)\) when DNS is used. Obviously, the more CNS is applicable, the more NS exploration is computationally effective. We show some experimental evidence that CNS is often applicable. Consider the exploration of an MDP with \(N = 3\) objectives in figure 2, where
the Pareto vertices are numbered with the same order they are explored by the algorithm, starting from vertex 1. Vertices 2 and 3 are obtained using DNS, vertex 4 is neighbour of the known vertices 1 and 3, and thus CNS is applied. Vertex 5 still requires DNS, but for the following 8 vertices (i.e. until vertex 13), CNS is applied. Once the search is completed, Direct Neighbour search is performed 5 times, whereas CNS is performed 12 times. Such a difference tends to increase dramatically with the size of the MDP. For the exploration shown in figure 3, the DNS is performed 44 time, whereas CNS is performed 767 times. Table 2 shows that the dominance of CNS on DNS also grows with the number of objectives $N$; the number of times when DNS is run seem stationary, or even decreasing, as $N$ increases. Intuitively, CNS never runs when there are only $N = 2$ objectives.

| $|X| = 10$ | $|A| = 5$ |
|---|---|---|---|
| $N$ | vertices | edges | DNS ( % ) | CNS ( % ) |
| 2 | 9 | 8 | 8 ( 100.0 % ) | 0 ( 0.0 % ) |
| 3 | 72 | 127 | 16 ( 22.5 % ) | 55 ( 77.5 % ) |
| 4 | 246 | 625 | 14 ( 5.7 % ) | 231 ( 94.3 % ) |
| 5 | 405 | 1,232 | 14 ( 3.5 % ) | 390 ( 96.5 % ) |

| $|X| = 20$ | $|A| = 10$ |
|---|---|---|---|
| $N$ | vertices | edges | DNS ( % ) | CNS ( % ) |
| 2 | 25 | 24 | 24 ( 100.0 % ) | 0 ( 0.0 % ) |
| 3 | 418 | 812 | 26 ( 6.2 % ) | 391 ( 93.8 % ) |
| 4 | 5,004 | 14,748 | 26 ( 0.5 % ) | 4,977 ( 99.5 % ) |
| 5 | 34,742 | 134,292 | 20 ( 0.1 % ) | 34,721 ( 99.9 % ) |

Table 2 Examples of NS exploration for different MDPs with different number of objectives $N$, highlighting the number of 0-Bellman submodels and cost-to-go functions, which have been obtained using DNS and CNS.
4 Summary and Conclusions

The methodology herein presented is an algorithm called Neighbour Search (NS), to construct the whole Pareto set of a convex polytope with $N$ dimensions, by means of collection of all Pareto vertices and Pareto edges, and to obtain the corresponding Pareto submodels. Neighbour Search is an iterative procedure; each iteration consists in two steps: Edges Exploration and Neighbour Detection. Edges Exploration takes a Pareto vertex and determines all Pareto edges connecting such a Pareto vertex to its neighbours. Each neighbour is again a Pareto vertex that is obtained by Neighbour Detection. The procedure continues until all Pareto vertices are explored. Neighbour Search is firstly applied to a generic multi-objective problem, where a Decision Maker has a finite set of solutions and each solution is evaluated with respect of $N$ performance criteria. Secondly, NS is applied to MDPs with $N$ discounted objectives. Novel numeric techniques are herein developed to effectively adapt Edges Exploration and Neighbour Detection to the MDPs characteristics. Edges Exploration consists of solving a problem of redundancy removal for systems of linear inequalities, taking time $K \cdot \text{LP}(N, |A(X)|)$, where $K$ is the number of neighbours of a vertex, and $|A(X)|$ is the size of the MDP. Neighbour Detection is performed either in time $\text{LP}(|X|, |A^e(s^*, X)|)$ by Direct Neighbour Search (DNS), or in time $O(N |A(X)|)$ by Cross Neighbour Search (CNS). The former requires the Bellman equation to be solved, even though with a reduced action set: $A^e(s^*, x) \subset A(x)$, $\forall x \in X$. The latter does not require the Bellman equation to be solved, and is computationally more efficient than DNS. However, CNS requires conditions that are not always fulfilled, whereas DNS is always applicable. Experimental results suggest that conditions for CNS to be applicable are actually satisfied for the most of NS iterations. In particular, CNS feasibility seems to grow with the size of the MDP and with the number of objectives $N$.

A Proofs

Some proof is based on the following theorem

**Theorem 1** The Rouché-Capelli theorem. Given the matrices:

$$
A = \begin{bmatrix}
   a_1^1 & \cdots & a_1^N \\
   \vdots & \ddots & \vdots \\
   a_M^1 & \cdots & a_M^N
\end{bmatrix}, \quad
b = \begin{bmatrix}
   b_1 \\
   \vdots \\
   b_M
\end{bmatrix}, \quad
A|b = \begin{bmatrix}
   a_1^1 & \cdots & a_1^N & b_1 \\
   \vdots & \ddots & \vdots & \vdots \\
   a_M^1 & \cdots & a_M^N & b_M
\end{bmatrix}
$$

the solution of a linear system of equations $A \cdot [x_1, \ldots, x_N]^T = b$ 1) exists if $\text{rank}(A) = \text{rank}(A|b)$, and 2) it is a linear subspace of $\mathbb{R}^N$ with dimension equal to $N - \text{rank}(A)$. A system of linear equations is homogeneous if all constant terms are zero: $b_1 = b_2 = \ldots = b_M = 0$. Every homogeneous system has at least one solution, known as the zero solution (or trivial solution), which is obtained by assigning the value of zero to each of the variables: $x_1 = x_2 = \ldots = x_N = 0$.

**Proof (Proposition 1)** The reader is referred to paper Dorini et al. [2007], where Proposition 1, makes the same statement, for the more generic case in which $\mathbb{R}^N$ is partially ordered, with
respect to a cone \( C \), such that 1) the closure \( \text{cl}(C) \) is a polyhedral cone, and 2) \( \text{cl}(C) = L \), is a non-empty linear subspace. Those conditions are called conditions \( C \). The concept of dominance adopted throughout this manuscript corresponds to a partial order generated by a standard cone, that fulfills conditions \( C \).

**Proof (Proposition 2)** Let \( S^* \subseteq S \) be the set generating \( F \) as the convex hull \( F = \text{conv}(S^*) \).

**Part a.** Since \( F \) is a \( k \)-face of \( D \), then we can collect no more than \( k+1 \) vectors \( s^1, \ldots, s^{k+1} \in S^* \) that are linearly independent. Every vector \( w \in W(S^*) \) must hence fulfil conditions

\[
I \quad \langle s^1, w \rangle = \langle s^2, w \rangle = \cdots = \langle s^{k+1}, w \rangle = \min_{w \in S^*} (s, w) = \min_{d \in D} (d, w).
\]

\[
II \quad \sum_{i=1}^{N} w_i = 1.
\]

\[
III \quad w_i > 0, i = 1, \ldots, N.
\]

If we put together points I and II, we obtain a system of linear inequations in the form

\[
A \cdot [w_1, \ldots, w_N]^T = b,
\]

where

\[
A = \begin{bmatrix}
s_1^1 - s_1^{k+1} & \cdots & s_1^1 - s_N^{k+1} \\
\vdots & \ddots & \vdots \\
s_1^1 - s_1^{k+1} & \cdots & s_N^1 - s_N^{k+1} \\
1 & \cdots & 1
\end{bmatrix}
\]

\[
b = \begin{bmatrix}
0 \\
\vdots \\
0 \\
1
\end{bmatrix}
\]

Since \( \text{rank}(A) = \text{rank}(A|b) = k+1 \), according to theorem 1, the space fulfilling conditions I and II has dimension equal to \( N - k - 1 \). Since condition III represents a non-empty polyhedron with dimension equal to \( N \), and since \( W(F) \) is bounded, we conclude that \( W(F) \) is a \((N-k-1)\)-polytope.

**Part b.** Consider the face \( F^w \) of \( D \) supported by the hyperplane \( H^w \) of equation 7; we have that:

\[
-F \subseteq F^w \quad \text{for every } w \in W(F).
\]

\[
-F = F^w \quad \text{for some } w \in W(F).
\]

Thus, \( F = \bigcap_{w \in W(F)} F^w = \bigcap_{w \in W(F)} H^w = \bigcap \text{aff} (F); \) thus the dimension of \( F \) is the dimension of \( \text{aff} (F) \). Since \( W(F) \) is a \((N-k-1)\)-polytope, then we can collect no more than \( N - k \) vectors \( w_1, w_2, \ldots, w_{N-k} \in W(F) \) that are linearly independent. Based on this, we have that \( \text{aff} (F) = \bigcap_{w \in W(F)} H^w = \bigcap_{k=1}^{N-k} H^w \), and this can be expressed as a system of linear inequalities in the form \( G \cdot [x_1, \cdots, x_N]^T = h \), where

\[
G = \begin{bmatrix}
w_1^1 & \cdots & w_N^1 \\
\vdots & \ddots & \vdots \\
w_1^{N-k} & \cdots & w_N^{N-k}
\end{bmatrix}
\]

\[
h = \begin{bmatrix}
\min_{d \in D} (d, w_1^1) \\
\vdots \\
\min_{d \in D} (d, w_{N-k}^1)
\end{bmatrix}
\]

Since \( \text{rank}(G) = \text{rank}(G|h) = N-k \), according to theorem 1, the dimension of \( \text{aff} (F) \) is equal to \( N - (N-k) = k \).

**Proof (Proposition 3)** Since \( s^{*} \in S^* \) (and \( s^{*} \in F \) then \( W(F) \subseteq W(s^*) \) (this follows from the definition of preference set, definition 7). A vector \( w \in W(s^*) \) is preference vector of \( S^* \) iff it fulfills the two conditions:

\[
\langle s^{*} - s, w \rangle = 0 \text{ iff } s \in S^* \setminus \{s^*\} \quad (31)
\]

\[
\langle s^{*} - s, w \rangle \leq 0 \text{ iff } s \notin S^* \quad (32)
\]

We notice that condition 32 is already guarantee by the conditions of membership of \( W(s^*) \). We use condition 31 to define \( W(F) \):

\[
W(F) = W(s^{*}) \cap \bigcap_{s \in S^*(s^{*})} H(s)
\]

where \( H(s) = \{ w \in \mathbb{R}^N | (s^{*} - s, w) = 0 \} \)

(33)
hence \( W(F) = W(S^*) \) is given by the intersection of \( W(s^*) \) with some hyperplanes. Such an intersection cannot contain any vector \( \hat{w} \in \text{ri}(W(s^*)) \), as this would imply the strict inequality \( (\hat{w}^* - s, \hat{w}) < 0 \) for every \( s \in S^\infty \setminus \{s^*\} \) which is not allowed (see definition 2).

We conclude that \( W(F) \) is a face of \( W(s^*) \), which is supported by the hyperplane resulting from the intersection \( \bigcap_{s \in S^\infty \setminus \{s^*\}} H(s) \) of equation 33. The vertex \( s^* \) is a face with dimension \( k = 0 \), then \( W(s^*) \) is a polytope with dimension \( N - k - 1 = N - 1 \); the edge \( F \) is a face with dimension \( k = 1 \), then \( W(F) \) is a polytope with dimension \( N - k - 1 = N - 2 \) (Proposition 2).

We conclude that \( W(F) \) is \( (N-2) \)-face of \( W(s^*) \), hence a facet, thus we have proved point i). Each hyperplane \( H(s), s \in S^\infty \setminus \{s^*\} \) supports \( W(s^*) \) on a face; hence facet \( W(F) \) is the intersection of faces. Intersection of two or more faces cannot be a facet unless 1) the faces are all facets and 2) they coincide; we conclude that each hyperplane \( H(s), s \in S^\infty \setminus \{s^*\} \) supports \( W(s^*) \) on \( W(F) \):

\[
W(F) = W(s^*) \cap H(s)
\]

this proves point ii).

**Proof (Proposition 5)** Let \( A^*(\cdot) \) be the action set of \( M^* \). Take an arbitrary vector \( w \in W(F) \), and let us consider two cases separately.

**Case** \( w \in W(F) \): Since \( \varphi^*(x) \in A^*(x), \forall x \in X \) then the same minimum of the Bellman equation 17 must be attained by the action \( a = \varphi^*(x) \)

\[
v^w(x) = \langle g(x, \varphi^*(x)), w \rangle + \beta \sum_{y \in X} p(y|x, \varphi^*(x)) \cdot v^w(y)
\]

for every \( x \in X \). Take now the cost-to-go as defined in expression 19, and perform the scalar product with \( w \) on both sides

\[
\langle J(\varphi^*, x), w \rangle = \langle g(x, \varphi^*(x)), w \rangle + \beta \sum_{y \in X} p(y|x, \varphi^*(x)) \cdot J(\varphi^*, y), w \rangle \quad x \in X
\]

by comparing with equation 34, considering that Bellman equation has unique solution (see Bertsekas [2000]), we conclude that \( v^w(x) = (J(\varphi^*, x), w) \), for every \( x \in X \). This proves point i.

**Case** \( w \notin W(F) \): Take any two stationary non-randomized policies \( \varphi^* \in M^* \), and \( \hat{\varphi} \in M^\infty \), generating the points \( V(\varphi^*) = s^* \neq \hat{s} = V(\hat{\varphi}) \). Since \( w \in W(\hat{s}) \) and \( w \notin W(s^*) \), then

\[
\langle \sum_{s \in X} J(\varphi^*, x), w \rangle = \langle s^*, w \rangle > \langle \hat{s}, w \rangle = \sum_{s \in X} v^w(x) \mu(x)
\]

and thus

\[
\sum_{x \in X} [(J(\varphi^*, x), w) - v^w(x)] \mu(x) > 0
\]

which implies point ii.

**Proof (Proposition 6)** Based on proposition 5, take a vector \( w \in W(s^*) \), we have that

\[
v^w(x) = \langle J(\varphi^*, x), w \rangle = \langle J(\varphi^*, x), w \rangle, \forall x \in X \). Since \( W(s^*) \) is a \( (N-1) \)-polytope (proposition 2), then we can collect no more than \( N \) vectors \( w^1, \ldots, w^N \in W(s^*) \) that are linearly independent. For every state \( x \in X \), the solution system of linear equations

\[
\begin{pmatrix}
  w_1^1 & \cdots & w_N^1 \\
  \vdots & \ddots & \vdots \\
  w_1^N & \cdots & w_N^N
\end{pmatrix}
\begin{pmatrix}
  J_1(\varphi^1, x) - J_1(\varphi^2, x) \\
  \vdots \\
  J_N(\varphi^1, x) - J_N(\varphi^2, x)
\end{pmatrix}
= \begin{pmatrix}
  0 \\
  \vdots \\
  0
\end{pmatrix}
\]

is a linear subspace of \( \mathbb{R}^N \) with dimension equal to \( N - \text{rank}(A) = N - \text{rank}(A|b) = 0 \). Since the system is admissible and homogeneous, the zero solution is the only possible solution (theorem 1), i.e. \( J_i(\varphi^1, x) - J_i(\varphi^2, x) = 0 \), \( i = 1, \ldots, N \).
Proof (Proposition 7) By combining Bellman equation 17 with the statement of proposition 5, we obtain that a vector \( w \in W \) is a preference vector \( w \in W(s^*) \) if and only if the following relationship
\[
\min_{a \in A(x)} \left[ (g(x, a), w) + \beta \sum_{y \in X} p(y | x, a) \langle J(s^*, y), w \rangle \right] = \langle J(s^*, x), w \rangle
\]
holds for every state \( x \in X \). This can be re-written by means of Q-function:
\[
\min_{a \in A(x)} \left[ (Q(s^*, x, a), w) \right] = \langle Q(s^*, x, \varphi^*(x)), w \rangle \quad x \in X
\]
Since such a minimum is always attained by all actions \( a \in A(s^*, x) \), for all \( x \in X \), then expression \( (Q(s^*, x, \varphi^*(x)) - Q(s^*, x, a), w) \leq 0 \) holds for every \( x \in X \) and for every \( a \in A(x) \setminus A(s^*, x) \).

Proof (Proposition 8) Point i) is the same as in Proposition 3. According to definition 10, given a vector \( w \in W(s^*) \), the action set \( A^w(\cdot) \) of submodel \( M^w \) is defined in equation 18. Equation 18 can be re-expressed by means of the Q-function:
\[
A^w(x) = \arg \min_{a \in A(x)} \left[ (Q(s^*, x, a), w) \right] \quad x \in X
\]
equivalent to equation
\[
A^w(x) = \{ a \in A(x) \mid \langle Q(s^*, x, \varphi^*(x)) - Q(s^*, x, a), w \rangle = 0 \} \quad x \in X
\]
We observe that \( W(F) \subseteq W(s^*) \), as \( s^* \subseteq S^W(s^*) \) (and \( s^* \in F \)). A vector \( w \in W(s^*) \) is preference vector of \( S^W(s^*) \) iff \( A^w(s^*, x) \subseteq A^w(x) \forall x \in X \), namely iff it fulfills the two conditions:
\[
\langle Q(s^*, x, \varphi^*(x)) - Q(s^*, x, a), w \rangle = 0 \text{ iff } x \in X, a \in A^w(s^*, x) \setminus A(s^*, x)
\]
\[
\langle Q(s^*, x, \varphi^*(x)) - Q(s^*, x, a), w \rangle \leq 0 \text{ iff } x \in X, a \notin A^w(s^*, x)
\]
We notice that condition 37 is already guaranteed by the conditions of membership of \( W(s^*) \). We use condition 37 to define \( W(F) \):
\[
W(F) = W(s^*) \cap \bigcap_{(x,a)} \{ w \in \mathbb{R}^N \mid \langle Q(s^*, x, \varphi^*(x)) - Q(s^*, x, a), w \rangle = 0 \}
\]
\[
\text{where } (x,a) \in X \times A \mid x \in X, a \in A^w(s^*, x) \setminus A(s^*, x)
\]
hence \( W(F) = W(S^W(s^*)) \) is given by the intersection of \( W(s^*) \) with some hyperplanes. Such an intersection is a facet (Proposition 3). Each hyperplane of equation 38 supports \( W(F) \) on a face; hence facet \( W(F) \) is the intersection of faces. Intersection of two or more faces cannot be a facet unless 1) the faces are all facets and 2) they coincide; we conclude that each hyperplane \( H(x, a) \) such that \( x \in X, a \in A^w(s^*, x) \setminus A(s^*, x) \) supports \( W(s^*) \) on \( W(F) \):
\[
W(F) = W(s^*) \cap H(x, a)
\]
this proves point ii).

Proof (Proposition 9) Point i: As described in step 3 of algorithm 1, the neighbour \( s^w \) can be found by taking an arbitrary point \( \tilde{s} \in S^W(s^*) \) \( \setminus \{ s^* \} \) and solve the problem \( s^w = \arg \min_{s \in S^W(s^*)} (s, a) \), where \( a \) can be either \( s^* - \tilde{s} \) or any other vector with same direction and orientation. According to proposition 3, point \( \tilde{s} \) defines a hyperplane \( \hat{H}(\tilde{s}) \)
\[
H(\tilde{s}) = \left\{ w \in \mathbb{R}^N \mid (s^* - \tilde{s}, w) = 0 \right\}
\]
supporting $W(s^*)$ on $W(F)$. According to proposition 8, set $W(F)$ is also supported by a hyperplane

$$H \{ x, a' \} = \left\{ w \in \mathbb{R}^N \mid \langle Q(s^*, x, a'), Q(s^*, x, a^\prime) \rangle, w \right\} = 0$$

defined by any arbitrary $\tilde{x}, a^1, a^\prime$ such that $\tilde{x} \in X$, $a^1 \in A(s^*, \tilde{x})$ and $a^\prime \in A(s^*, \tilde{x})$. Now, both $H(\bar{z})$ and $H(\bar{x}, a')$ are spaces with dimension $N - 1$; they both contain $W(F)$, which is a $(N - 2)$-polytope; they both contain the origin (which does not belong to $W(F)$, as it does not even belong to $W$). We conclude that $H(\bar{z})$ and $H(\bar{x}, a')$ are the same hyperplane, therefore $s^* - \bar{z} = Q(s^*, x, a') - Q(s^*, x, a^\prime)$ are parallel. Furthermore, for any vector $w \in \mathbb{R}^N$ we have that $w \in W(s^*)$ if and only if $\langle s^* - \bar{z}, w \rangle \leq 0$. According to proposition 7 we also have that $w \in W(s^*)$ if and only if $\langle Q(s^*, x, a') - Q(s^*, x, a^\prime), w \rangle \leq 0$. In other words $s^* - \bar{z}$ and $Q(s^*, x, a') - Q(s^*, x, a^\prime)$ define the same halfspace, and thus they are parallel and the same orientation. We can solve problem $s^n = \arg \min_{s \in \mathbb{R}^N} \langle s, \alpha \rangle$ by setting

$$\alpha = Q(s^*, x, a') - Q(s^*, x, a^\prime)$$

and solving the Bellman equation 26.

Proof: By combining propositions 5 and 6, take a vector $w \in W(F)$, we have that

$$w(x) = (J(s^*, x), w) = \langle J(s^*, x), w \rangle, \forall x \in X.$$  Since $W(F)$ is a $(N - 2)$-polytope (proposition 2), then we can collect no more than $N - 1$ vectors $w^1, \ldots, w^{N-1} \in W(F)$ that are linearly independent. For every state $x \in X$, the solution system of linear equations

$$\begin{bmatrix} w_1^1 & \cdots & w_1^N \\ \vdots & \ddots & \vdots \\ w_{N-1}^1 & \cdots & w_{N-1}^N \end{bmatrix} \begin{bmatrix} J_1(s^*, x) - J_1(s^*, x) \\ \vdots \\ J_N(s^*, x) - J_N(s^*, x) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

is a linear subspace of $\mathbb{R}^N$ with dimension equal to $N - \text{rank}(A) = N - \text{rank}(A|b) = 1$, hence it is a straight line; hence then for every $x \in X$

$$J(s^*, x) - J(s^*, x) = \alpha' \langle x \rangle$$

where $\alpha'(x) \in \mathbb{R}$, and $\alpha'$ is a vector of $\mathbb{R}^N$, such that $\alpha' \perp W(F)$. Since $Q(s^*, x, a') - Q(s^*, x, a^\prime) \perp W(F)$ then we can assign $\alpha := Q(s^*, x, a') - Q(s^*, x, a^\prime)$. We combine this with propositions 5 and 6, and we obtain for every $x \in X$

$$\hat{v}^\alpha(x) = \langle J(s^*, x), \alpha \rangle = \langle J(s^*, x), \alpha \rangle + \alpha' \langle x \rangle$$

then

$$\sum_{y \in X} p(y|x, a) \hat{v}^\alpha(y) = \sum_{y \in X} p(y|x, a) \langle J(s^*, y), \alpha \rangle + \alpha \sum_{y \in X} p(y|x, a) \alpha' \langle y \rangle$$

and finally

$$\langle g(x, a), \alpha \rangle + \beta \sum_{y \in X} p(y|x, a) \hat{v}^\alpha(y) = \cdots$$

$$\cdots = \langle g(x, a), \alpha \rangle + \beta \sum_{y \in X} p(y|x, a) \langle J(s^*, y), \alpha \rangle + \alpha \beta \sum_{y \in X} p(y|x, a) \alpha' \langle y \rangle$$

$$\cdots = \langle Q(s^*, x, a), \alpha \rangle + \alpha \langle \alpha \rangle \tau(x, y) = \langle Q(s^*, x, a), \alpha \rangle$$

References


