

An Optimization Problem in Adaptive Virtual Environments

Department of Computer Science, Northwestern University

1 Dynamic adaptation problem in virtual execution environments

A virtual execution environment consisting of virtual machines (VMs) interconnected with virtual networks provides opportunities to dynamically optimize, at run-time, the performance of existing, *unmodified* distributed applications without any user or programmer intervention. Along with resource monitoring and inference and application-independent adaptation mechanisms, efficient adaptation algorithms are key to the success of such an effort. Here, we formalize the adaptation problem, prove that it is NP-hard.

1.1 Problem formulation

We formalize the problem as follows:

Problem 1 (Generic Adaptation Problem In Virtual Execution Environments (GAPVEE))

INPUT:

- A directed graph $G = (H, E)$
- A function $\text{bw} : E \rightarrow \mathbb{R}$
- A function $\text{lat} : E \rightarrow \mathbb{R}$
- A function $\text{compute} : H \rightarrow \mathbb{R}$
- A function $\text{size} : H \rightarrow \mathbb{R}$
- A set, $\text{VM} = (\text{vm}_1, \text{vm}_2 \dots \text{vm}_n), n \in \mathbb{N}$
- A function $\text{vm_compute} : \text{VM} \rightarrow \mathbb{R}$
- A function $\text{vm_size} : \text{VM} \rightarrow \mathbb{R}$
- A set of ordered 4-tuples $\mathcal{A} = \{(s_i, d_i, b_i, l_i) \mid s_i, d_i \in \text{VM}; b_i, l_i \in \mathbb{R}; i = 1, \dots, m\}$
- A set of ordered pairs $\mathcal{M} = \{(\text{vm}_i, h_i) \mid \text{vm}_i \in \text{VM}, h_i \in H; i = 1, 2 \dots r, r \leq n\}$

OUTPUT: $\text{vmap} : VM \rightarrow H$ and $R : \mathcal{A} \rightarrow \mathcal{P}$ such that

- $\sum_{\text{vmap}(\text{vm})=h} (\text{vm_compute}(\text{vm})) \leq \text{compute}(h), \forall h \in H$
- $\sum_{\text{vmap}(\text{vm})=h} (\text{vm_size}(\text{vm})) \leq \text{size}(h), \forall h \in H$
- $h_i = \text{vmap}(\text{vm}_i), \forall M_i = (\text{vm}_i, h_i) \in \mathcal{M}$
- $(\text{bw}_e - \sum_{e \in R(A_i)} b_i) \geq 0, \forall e \in E$
- $(\sum_{e \in R(A_i)} \text{lat}_e) \leq l_i, \forall e \in E$
- $\sum_{i=1}^m (\min_{e \in R(A_i)} \{\text{rc}_e\})$, where $\text{rc}_e = (\text{bw}_e - \sum_{e \in R(A_i)} b_i)$, is maximized

2 A special case of the adaptation problem

The generic adaptation problem seeks a mapping, vmap from VMs to hosts and routing, R of VM traffic over the overlay network, G . To establish the hardness of the problem, we consider a special case of the problem wherein all the VM to host mappings are constrained by the ordered pairs \mathcal{M} and latency demands are dropped, leaving us only with the routing problem.

Since the mappings are pre-defined, we can formulate the problem in terms of only the hosts and exclude all VMs. Also, as the latency demands have been dropped, the application 4-tuple reduces to 3-tuple, $A_i = (s_i, d_i, b_i)$, $s_i, d_i \in H$, $b_i \in \mathbb{R}$, $i = 1, 2 \dots m$. Notice that now $s_i, d_i \in H$ as VM to host mappings are fixed and VMs are synonymous with the hosts that they are mapped to.

2.1 Problem formulation

We formalize the problem as follows:

Problem 2 (Routing Problem In Virtual Execution Environments (RPVEE))

INPUT:

- A directed graph $G = (H, E)$
- A function $\text{bw} : E \rightarrow \mathbb{R}$
- A set of ordered 3-tuples $\mathcal{A} = \{(s_i, d_i, b_i) \mid s_i, d_i \in H; b_i \in \mathbb{R}; i = 1, \dots, m\}$

OUTPUT: $R : \mathcal{A} \rightarrow \mathcal{P}$ such that

- $(\text{bw}_e) - (\sum_{e \in R(A_i)} b_i) \geq 0, \forall e \in E$,
- $\sum_{i=1}^m (\min_{e \in R(A_i)} \{\text{rc}_e\})$, where $\text{rc}_e = (\text{bw}_e - \sum_{e \in R(A_i)} b_i)$, is maximized

3 Analysis

3.1 Analysis of RPVEE

The NP-hardness for the problem is established by reduction from the Edge Disjoint Path Problem (EDPP) which has been shown to be NP-complete [1]. To prove the NP-hardness of RPVEE we take any arbitrary instance of EDPP and convert it to a particular instance of the decision version of RPVEE, RPVEED. We then show that EDPP will have a “Yes” solution if and only if we have a solution to the RPVEED.

This can also be stated as, we consider an arbitrary instance of EDPP and convert it to a particular instance of of RPVEED. We show that if a polynomial-time solution exists for the RPVEED, then it will solve any arbitrary instance of EDPP. This boils down to proving the following

- A polynomial-time solution to the particular instance of RPVEED will also be a polynomial-time solution to any arbitrary instance of EDPP.
- If no polynomial-time solution exists to RPVEED then no polynomial-time solution exists for any arbitrary instance of EDPP.

The edge disjoint path problem (EDPP) is specified as:

Problem 3 (Edge Disjoint Path Problem (EDPP))

INPUT:

- A graph $G = (H, E)$, $|H| = p$, $|E| = q$
- A set of 2-tuples $\mathcal{A} = \{(s_i, d_i) \mid s_i, d_i \in H; i = 1, \dots, m\}$

OUTPUT:

- $\forall (s_i, d_i) \in \mathcal{A}$ to determine if their exist edge disjoint paths in $G = (H, E)$

The decision version of RPVEE (RPVEED) is specified as:

Problem 4 (Decision version of Routing Problem In Virtual Execution Environments (RPVEED))

INPUT:

- A directed graph $G = (H, E)$
- A function $\text{bw} : E \rightarrow \mathbb{R}$
- A set of ordered 3-tuples $\mathcal{A} = \{(s_i, d_i, b_i) \mid s_i, d_i \in H; b_i \in \mathbb{R}; i = 1, \dots, m\}$

OUTPUT: $R : \mathcal{A} \rightarrow \mathcal{P}$ such that

- $(\text{bw}_e) - (\sum_{e \in R(A_i)} b_i) \geq 0, \forall e \in E,$
- $\sum_{i=1}^m (\min_{e \in R(A_i)} \{\text{rc}_e\}), \text{ where } \text{rc}_e = (\text{bw}_e - \sum_{e \in R(A_i)} b_i), \geq k, k \in \mathbb{R}$

Lemma 1 *If a polynomial-time solution exists for RPVEED, then a solution exists for any arbitrary instance of EDPP.*

Proof The existence of a polynomial time solution to the routing problem implies that $\sum_{i=1}^m (\min_{e \in R(A_i)} \{rc_e\})$, where $rc_e = (bw_e - \sum_{e \in R(A_i)} b_i)$, $\geq k, k \in \mathbb{R}$. Since the contributions from any path can be either 0 or ϵ , it is implied that all 3-tuples are mapped such that only edges with weights $(i + \epsilon)$ are part of the paths, i.e. edges $\in E$. Further, each of these edges can be part of only a single path since all demands, d_i , are 1 and all the edge weights are < 2 . This implies that each of the mapped paths are edge disjoint. This proves that the existence of a polynomial-time solution to the particular instance of our problem implies a solution to any arbitrary instance of the edge disjoint path problem.

Lemma 2 *If no polynomial-time solution exists to RPVEED then it implies that no polynomial-time solution exists for any arbitrary instance of EDPP.*

Proof We will prove this by contradiction. Lets assume a polynomial-time solution exists for the edge disjoint path problem, while no polynomial-time solution exists for the particular instance of the routing problem.

The capacities for all the edges participating in the paths are $(1 + \epsilon)$ and each edge participates only in a single path. The residual bottleneck bandwidth for each mapped path will be ϵ . Since we would have successfully mapped k such paths, the sum of the residual bottleneck bandwidths for the paths would be $k \cdot \epsilon$. This implies that there exists a polynomial-time solution to the particular instance of the routing problem, thus completing the proof by contradiction.

Theorem 1 *RPVEE is NP-hard.*

Proof For reducing EDPP to an instance of RPVEED, construct a directed graph $G' = (V, E')$ where $bw((u, v)) = 1 + \epsilon$ if $(u, v) \in E$ and $bw((u, v)) = 1$ if $(u, v) \ni E$. Further for all $(s_i, t_i) \in S$, let $(s_i, d_i, 1) \in \mathcal{A}$. It is obvious that this reduction can be done in $O(n^2)$.

Since we can reduce, in polynomial-time, a NP-complete problem, EDPP, to RPVEED and by Lemma 1 and Lemma 2, we have proved that the transformation works, we have proved RPVEED to be NP-hard. This proves that the optimization version of RPVEED, RPVEE is NP-hard.

3.2 Analysis of GAPVEE

The NP-hardness for the problem is established by reduction from RPVEED, which has already been shown to be NP-complete. To prove the NP-hardness of GAPVEE we take any arbitrary instance of RPVEED and convert it to a particular instance of the decision version of GAPVEE, GAPVEED. We then show that RPVEED will have a “Yes” solution if and only if we have a solution to the GAPVEED.

The decision version of GAPVEE (GAPVEED) is state as follows

Problem 5 (Decision version of Generic Adaptation Problem In Virtual Execution Environments (GAPVEED))

INPUT:

- A directed graph $G = (H, E)$
- A function $\text{bw} : E \rightarrow \mathbb{R}$
- A function $\text{lat} : E \rightarrow \mathbb{R}$
- A function $\text{compute} : H \rightarrow \mathbb{R}$
- A function $\text{size} : H \rightarrow \mathbb{R}$
- A set, $\text{VM} = (\text{vm}_1, \text{vm}_2 \dots \text{vm}_n), n \in \mathbb{N}$
- A function $\text{vm_compute} : \text{VM} \rightarrow \mathbb{R}$
- A function $\text{vm_size} : \text{VM} \rightarrow \mathbb{R}$
- A set of ordered 4-tuples $\mathcal{A} = \{(s_i, d_i, b_i, l_i) \mid s_i, d_i \in \text{VM}; b_i, l_i \in \mathbb{R}; i = 1, \dots, m\}$
- A set of ordered pairs $\mathcal{M} = \{(\text{vm}_i, h_i) \mid \text{vm}_i \in \text{VM}, h_i \in H; i = 1, 2 \dots r, r \leq n\}$

OUTPUT: $\text{vmap} : \text{VM} \rightarrow H$ and $R : \mathcal{A} \rightarrow \mathcal{P}$ such that

- $\sum_{\text{vmap}(\text{vm})=h} (\text{vm_compute}(\text{vm})) \leq \text{compute}(h), \forall h \in H$
- $\sum_{\text{vmap}(\text{vm})=h} (\text{vm_size}(\text{vm})) \leq \text{size}(h), \forall h \in H$
- $h_i = \text{vmap}(\text{vm}_i), \forall M_i = (\text{vm}_i, h_i) \in \mathcal{M}$
- $(\text{bw}_e - \sum_{e \in R(A_i)} b_i) \geq 0, \forall e \in E$
- $(\sum_{e \in R(A_i)} \text{lat}_e) \leq l_i, \forall e \in E$
- $\sum_{i=1}^m (\min_{e \in R(A_i)} \{\text{rc}_e\}), \text{ where } \text{rc}_e = (\text{bw}_e - \sum_{e \in R(A_i)} b_i), \geq k, k \in \mathbb{R}$

Theorem 2 GAPVEE is NP-hard.

Proof For reducing RPVEED to an instance of GAPVEED, construct a directed graph $G = (H, E)$ where

- $\text{lat}((u, v)) = 0 \forall (u, v) \in E$
- $\text{compute}(h) = (n + \epsilon) \forall h \in H$
- $\text{size}(h) = (n + \epsilon) \forall h \in H$

Also introduce a set, $\text{VM} = (\text{vm}_1, \dots, \text{vm}_m)$, such that $\forall (s_i, d_i, b_i) \in \mathcal{A}$ (RPVEED), $s_i, d_i \in \text{VM}$. This would define the set of ordered pairs.

Further $\forall \text{vm} \in \text{VM}$

- $\text{vm_compute}(\text{vm}) = 1$
- $\text{vm_size}(\text{vm}) = 1$

Finally for all $(s_i, d_i, b_i) \in \mathcal{A}$ (RPVEED), let $(s_i, d_i, b_i, 1) \in \mathcal{A}$ (GAPVEED).

It is obvious that this reduction can be done in $O(n^2)$.

Since we can reduce, in polynomial-time, a NP-complete problem, RPVEED, to GAPVEED and since it is trivially clear that a polynomial-time solution to RPVEED will exist if and only if a polynomial-time solution to GAPVEED exists, we have proved GAPVEED to be NP-hard. This proves that the optimization version of GAPVEED, GAPVEE is NP-hard.

References

- [1] KARP, R. *Complexity of Computer Computations*. Miller, R.E. and Thatcher, J.W. (Eds.). Plenum Press, New York, 1972, ch. Reducibility among combinatorial problems, pp. 85–103.