International Journal of Foundations of Computer Science
Vol. 29, No. 7 (2018) 1231–1245
© World Scientific Publishing Company
DOI: 10.1142/S0129054118500314

World Scientific
www.worldscientific.com

The Computational Complexity of and Approximation Algorithms for Variants of the Component Selection Problem

Mostafa Nouri-Baygi

Department of Computer Engineering
Ferdowsi University of Mashhad
Mashhad, Iran
nouribaygi@um.ac.ir

Received 14 February 2017
Accepted 26 February 2018
Communicated by Mitsunori Ogihara

In the past decades, there has been a burst of activity to simplify implementation of complex software systems. The solution framework in software engineering community for this problem is called component-based software design (CBSD), whereas in the modeling and simulation community it is called composability. Composability is a complex feature due to the challenges of creating components, selecting combinations of components, and integrating the selected components.

In this paper, we address the challenge through the analysis of Component Selection (CS), the NP-complete process of selecting a minimal set of components to satisfy a set of objectives. Due to the computational complexity of CS, we consider approximation algorithms that make the component selection process practical. We define three variations of CS and present good approximation algorithms to find near optimal solutions. In spite of our creation of approximable variants of Component Selection, we prove that the general Component Selection problem is inapproximable.

Keywords: Component selection; approximation algorithms; NP-completeness; set cover; red-blue set cover.

1. Introduction

Over recent decades, one of the approaches used to overcome the highly increasing demand for new products was the concept of reusing. In software engineering, component-based software design considers building applications from existing components as a way of reusing rather than writing them from scratch, for example [4, 7, 16]. In simulation, composability is the ability to combine reusable simulation components to satisfy a set of user objectives. Composability is highly demanded for model and simulation developers because of the benefits afforded by reuse, for example [2, 17].

Component Selection (CS) is the problem of choosing the minimum number of components from a set of components such that their composition satisfies a set of objectives. Component Selection is an NP-complete [15] optimization problem, and has been formally shown to be embedded in composability [15]. For composability to be achievable in reasonable time, the selection of a set of components must be possible in polynomial time. Fox *et al.* [9] conjectured that in its general form CS is inapproximable.

Baker *et al.* [1] investigated CS and proposed a greedy algorithm and a simulated annealing algorithm to solve the problem. Fahmi and Choi [8] introduced case based reasoning in component selection and evaluated the approach. Khan and Mahmood [11] gave a component selection process that uses a signed graph and groups related goals into clusters, based on the usage, non-functional and threat dependencies.

In this paper we focus on CS and try to solve it approximately in different cases. The results we have obtained can be summarized as below:

- Proving that CS is inapproximable when compositions exhibit emergent or anti-emergent behavior, a conjecture which is suggested by Fox *et al.* [9].
- Proposing an approximation algorithm for CS, when each component has a unit cost and the number of emergent and non-emergent compositions is bounded. We also give an approximation ratio for the algorithm and a tight example that attains the ratio.
- Defining a new version of CS, in which each component has a real valued cost. We then present an approximation algorithm for the problem, prove an approximation ratio and show a tight example which produces that ratio.
- Defining a new version of CS, in which each component may have several types of cost, possibly common with other components. We then present an approximation algorithm for the problem with a provable approximation ratio.

This paper is an extended version of [12] in which all omitted proofs of theorems in that paper are presented completely and some recent related works are cited.

In the following we will examine component selection in three different cases. In Sec. 2, we will study the well-known problem of component selection with unit cost. Then in Sec. 3, we will extend the problem to the case in which each component has a real valued cost. In Sec. 4 the multi-cost component selection problem is defined and an approximation algorithm for it is presented. Finally, in Sec. 5 we will summarize the results of this paper.

2. Component Selection with Unit Cost

In this section we consider the simplest form of component selection, CSUC, in which adding each component charges a unit cost to the composition. Informally, the problem is to select and compose the minimum number of components from

a repository of components such that the composition will meet a given set of objectives.

Let $O = \{o_1, o_2, \ldots, o_n\}$ be a set of objectives and $C = \{c_1, c_2, \ldots, c_m\}$ be a set of components. A simulation system S is a subset of C, i.e. $S \subseteq C$. If $|S| > 1$ then S is a composition. Let $\circ$ denote the composition of components, e.g. $(c \circ c')$ is the composition of c and c'. Let $\vDash$ and $\nvDash$ denote *satisfies* and *does not satisfy* an objective respectively, e.g. $c \vDash o$ means c satisfies o and $c \nvDash o$ means c does not satisfy o. These two operators may also be used with compositions and sets of objectives and have the expected meanings, e.g. $c \circ c' \vDash o$ and $S \nvDash O$. A simulation system $S \vDash O$ if and only if for every $o_i \in O$, $S \vDash o_i$.

In some variants of CSUC, the set of objectives satisfied by a composition may not be the union of the objectives satisfied by the components individually. With regard to the set of objectives satisfied by a composition of a set of components there may be three different situations which can be seen in Table 1. Two of these situation (emergent and non-emergent) were introduced by Page and Opper [13] and the third one (anti-emergent) was later defined by Petty *et al.* [15]. Informally, if none of c, c' and $c \circ c'$ satisfy o, then the composition is *non-emergent*. If c and c' do not satisfy o but $c \circ c'$ satisfies o, then the composition is *emergent*. If one or both of c and c' satisfy o but $c \circ c'$ does not satisfy o, then the composition is *anti-emergent*. In Table 1 all different possible logical combinations of satisfaction of an objective by components have been shown.

Petty *et al.* [15] defined a general problem which subsumes all these different variations of the problem. They showed that even in the presence of an oracle function that can determine in one step which objectives are satisfied by a component or a composition, the problem of component selection with unit costs (CSUC) is NP-complete. The formal definition of the decision version of CSUC is as follows:

CSUC

Input: A set $C = \{c_1, c_2, \ldots, c_m\}$ of components, a set $O = \{o_1, o_2, \ldots, o_n\}$ of objectives, an oracle function $\sigma : 2^C \to 2^O$, and a positive integer $K \leq |C|$.

Question: Is there a composition $S \subseteq C$ such that $|S| \leq K$ and $O \subseteq \sigma(S)$?

Table 1. Different types of compositions.

Objective Satisfaction of Components		Objective Satisfaction of Composition	Composition Type
$c \vDash o$	$c' \vDash o$	$(c \circ c') \vDash o$	Non-emergent
$c \nvDash o$	$c' \vDash o$	$(c \circ c') \vDash o$	Non-emergent
$c \vDash o$	$c' \nvDash o$	$(c \circ c') \vDash o$	Non-emergent
$c \nvDash o$	$c' \nvDash o$	$(c \circ c') \nvDash o$	Non-emergent
$c \nvDash o$	$c' \nvDash o$	$(c \circ c') \vDash o$	Emergent
$c \vDash o$	$c' \vDash o$	$(c \circ c') \nvDash o$	Anti-emergent
$c \nvDash o$	$c' \vDash o$	$(c \circ c') \nvDash o$	Anti-emergent
$c \vDash o$	$c' \nvDash o$	$(c \circ c') \nvDash o$	Anti-emergent

Since this problem has been proved to be NP-complete, it is very unlikely to find a polynomial time algorithm to solve CSUC (unless P = NP). Therefore it is natural to look for an algorithm that approximately computes the minimum number of components needed to satisfy all the objectives in O.

Fox *et al.* [9] have demonstrated that the greedy algorithm is not an approximation algorithm for CSUC. Since the greedy algorithm is one of the best algorithms for *Set Cover*, which is very closely related to CSUC, they conjectured that there is no algorithm that can approximate CSUC. In the following two theorems, we prove that when the compositions have emergent or anti-emergent behavior, CSUC cannot be approximated.

Theorem 1. *CSUC with emergent compositions cannot be approximated to within* $o(\sqrt{m})$*, where* m *is the number of components.*

Proof. For the sake of contradiction, assume there exists a polynomial time approximation algorithm $\mathcal{A}$ for the problem with an approximation ratio $r \in o(\sqrt{m})$. Consider the following instance I: Let $m_1 = \frac{m}{r+1}$ and $m_2 = m - m_1$ and assume for each compositions $S \subset C$, where $|S| < m_1$, $\sigma(S) = \varnothing$, and for each $S \subseteq C$, where $|S| > m_2$, $\sigma(S) = O$. This implies that for each $c_i \in C$, $\sigma(\{c_i\}) = \varnothing$.

Let $\mathcal{A}$ be executed on I. We will show that there is a composition S_1 of cardinality m_1 such that for each $S_2 \supseteq S_1$ where $|S_2| \leq m_2$, $\mathcal{A}$ does not check $\sigma(S_2)$. If we prove the existence of such S_1, we could assume that for each $S \supseteq S_1$, $\sigma(S) = O$ and for all $S \not\supseteq S_1$ where $|S| \leq m_2$, $\sigma(S) = \varnothing$. In fact we imagine an adversary that chooses values of $\sigma(\cdot)$ for each composition to deceive $\mathcal{A}$ in such a way that there will not be any contradiction in previous queries, i.e. $\sigma(S) \subseteq \sigma(S')$ whenever $S \subseteq S'$. With this construction, the optimum answer is S_1 with m_1 components while the answer returned by $\mathcal{A}$ has more than m_2 components.

We call $S_1 \subseteq C$ of m_1 components a *good* composition if for all $S_2 \supseteq S_1$ where $|S_2| \leq m_2$, the algorithm $\mathcal{A}$ does not check the value of $\sigma(S_2)$, and otherwise a *bad* composition. Assume by contradiction that there does not exist such a good composition. Therefore for each composition S_1 with cardinality m_1, $\mathcal{A}$ must query the value of $\sigma(\cdot)$ for S_1 or one of its supersets of cardinality at most m_2. The number of candidates for good compositions is $\binom{m}{m_1}$. When $\mathcal{A}$ queries the value of $\sigma(S)$ for some composition S, where $m_1 \leq |S| \leq m_2$, it makes $\binom{|S|}{m_1}$ candidates of being good compositions bad. The maximum number achieved when $|S| = m_2$. Therefore, $\mathcal{A}$ must query $\sigma(\cdot)$ for at least

$$\begin{aligned}\frac{\binom{m}{m_1}}{\binom{m_2}{m_1}} &\geq \left(\frac{m}{m_2}\right)^{m_1} \\ &= \left(\frac{r+1}{r}\right)^{m/(r+1)} \\ &= \left(1+\frac{1}{r}\right)^{m/(r+1)}\end{aligned}$$

$$= \left((1 + \frac{1}{r})^{r+1} \right)^{m/(r+1)^2}$$
$$\geq e^{m/(r+1)^2}$$

compositions in polynomial time, which grows exponentially if $r \in o(\sqrt{m})$. The last inequality obtained from $(1 + 1/x)^{x+1} > e$, which holds for any positive x, where $e \approx 2.71828$ is the base of the natural logarithm.

This contradiction proves that there is at least one good composition. The existence of such a good composition shows that the approximation ratio of $\mathcal{A}$ is greater than $\frac{m_2}{m_1} = r$, which contradicts our first assumption. □

Theorem 2. *CSUC with anti-emergent compositions is not approximable.*

Proof. For the sake of contradiction, assume there exists a polynomial time approximation algorithm $\mathcal{A}$ for the problem. Consider the following instance I of the problem: Let $m = 2n$, where n is the number of objectives and m is the number of components. Each objective $o_i \in O$ is satisfied by components c_{2i} and c_{2i+1}. Therefore from each group $G_i = \{c_{2i}, c_{2i+1}\}$ at least one component must be selected in the final solution. Notice that the total number of candidate solutions is 2^n, which is not polynomial.

Consider an adversary that returns $\sigma(S) = \varnothing$ for each composition S queried by $\mathcal{A}$ among candidate solutions. Therefore $\mathcal{A}$ abandons further queries after a polynomial time and returns no result, but the optimal answer is a particular composition of n components selected by the adversary. In this instance of the problem, $\mathcal{A}$ cannot even find a possible composition of components that satisfy all the objectives. It follows that no approximation algorithm exists for CSUC with anti-emergent compositions. □

While we have proved that CSUC in its general form (if emergent or anti-emergent compositions exist) is not approximable, we can present an approximation algorithm for CSUC if the compositions exhibit only *non-emergent* behavior. We can use a greedy algorithm similar to the algorithm used for Set Cover and get an approximation ratio of $H(k) = \sum_{i=1}^{k} \frac{1}{i}$, where $k = \max\{|\sigma(\{c\})| : c \in C\}$. The greedy algorithm when CSUC only has non-emergent rules is shown in Algorithm 1. In the following theorem, it will be proved that the approximation ratio of the algorithm is $H(k)$.

Theorem 3. Greedy-CSUC *when σ has no emergent or anti-emergent behavior, approximates the solution with ratio $H(k) = \sum_{i=1}^{k} \frac{1}{i}$ where $k = \max\{|\sigma(\{c\})| : c \in C\}$.*

Proof. We prove this ratio with a transformation of the problem to Set Cover [5]. For clarity, we here state the formal definition of Set Cover.

Algorithm 1. The algorithm for CSUC with only non-emergent rules.

```
1: function GREEDY-CSUC(C, O, σ)
2:     U ← O                                   ▷ To be satisfied objectives
3:     S ← ∅                                   ▷ Selected components
4:     while U ≠ ∅ do
5:         Select a c_i ∈ C that maximizes |σ({c_i}) ∩ U|
6:         S ← S ∪ {c_i}                       ▷ Select the component
7:         U ← U − σ({c_i})                    ▷ Remove satisfied objectives
8:         return S
9:     end while
10: end function
```

Set Cover

Input: A set X, a family $\mathcal{F}$ of subsets of X, such that each element of X belongs to at least one subset in $\mathcal{F}$, i.e. $X = \bigcup_{S\in\mathcal{F}} S$.

Question: Find a subset $\mathcal{C} \subseteq \mathcal{F}$ of minimum size such that $X = \bigcup_{S\in\mathcal{C}} S$

A greedy algorithm similar to GREEDY-CSUC can approximate Set Cover (SC) with ratio $H(k)$ where k is the size of the largest set in $\mathcal{F}$ [10].

In order to prove the ratio for CSUC, we establish a one-to-one correspondence between instances of CSUC and SC. The optimum solution of SC also corresponds to the optimum solution of CSUC. The transformation is as follows. Let $X = O$. For each $c_i \in C$ create a set S_i containing each $o_j \in \sigma(\{c_i\})$ and add S_i to $\mathcal{F}$.

For each possible solution S for CSUC, $O \subseteq \sigma(S) = \bigcup_{c_i\in S} \sigma(\{c_i\})$ and so for the corresponding subfamily $\mathcal{C} \subseteq \mathcal{F}$, $X \subseteq \bigcup_{S\in\mathcal{C}} S$. The correspondence in the opposite direction is shown similarly. Since the cost of any solution $\mathcal{C}$ of SC is the number of sets in $\mathcal{C}$, which is equal to the number of components in the corresponding solution S of CSUC, the approximation ratio of the greedy algorithm holds for CSUC and is at most $H(k)$. □

We have to mention here that if the number of emergent and anti-emergent compositions are polynomially bounded and can be iterated by σ, CSUC can be approximated with the greedy algorithm. The trick is to treat each emergent or anti-emergent composition as a new component such that the satisfied objectives for that component is as the corresponding rule, and also the cost of the component is the number of components used inside that. We should also note that if these new components are selected, the inner components cannot be selected concurrently. By using the algorithm presented in Sec. 3, this problem can easily be approximated.

It is interesting to see that the approximation ratio is tight. We give an example, depicted in Fig. 1, which shows the tightness of this ratio. Consider the problem in which we have $k \cdot k!$ objectives, o_{ij} where $1 \le i \le k$ and $1 \le j \le k!$. For each r, $1 \le r \le k!$, for each s, $1 \le s \le k$, and for each t, $1 \le t \le k!(k - s + 1)$, we consider two groups of components c_1^r and c_2^{st}. Clearly the number of components

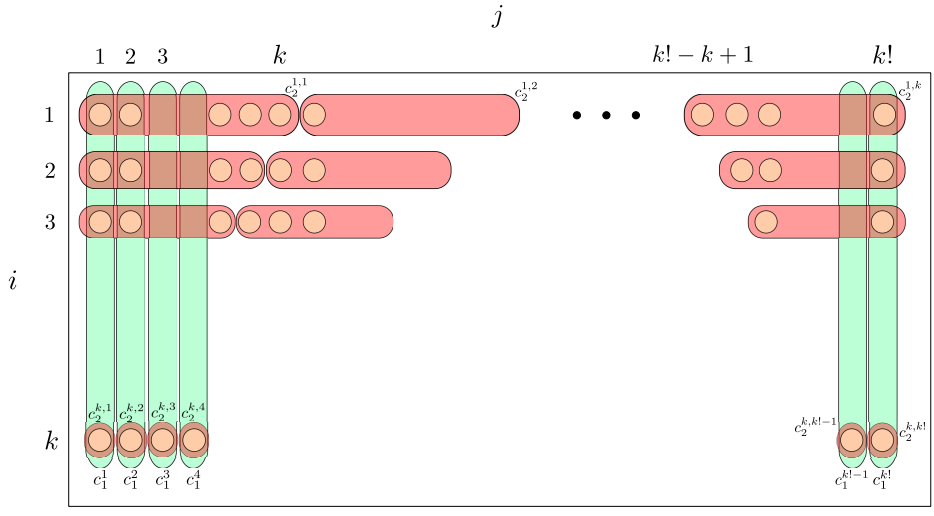

Fig. 1. An instance of CSUC that shows the approximation ratio of the greedy algorithm is tight. In the figure, the golden circles are objectives, the green bars are the components in the optimum solution, while the red bars are the components that may be returned by the greedy algorithm.

in the first group is $k!$ and in the second one is $k! \cdot (1 + 1/2 + \cdots + 1/k) = k! \cdot H(k)$. Each o_{ij} is satisfied by two components: c_1^j from the first group and $c_2^{i,\lfloor j/(k-i+1) \rfloor}$ from the second group. Therefore each component c_1^r satisfies k objectives, while each component c_2^{st} satisfies $k - s + 1$ objectives.

It is easy to see that the optimal solution is the set of components in the first group, but the greedy algorithm may return the set of components in the second group. The reason is that in the first $k!/k$ iterations of the **while** loop of Algorithm 1, both c_1^r and c_2^{1t} for $1 \leq r \leq k!$ and $1 \leq t \leq k!/k$ maximizes $|\sigma(\{c\}) \cap U|$ to k, so the algorithm may select the components in the second group. If the algorithm selects these components, in the next $k!/(k-1)$ iterations of the **while** loop, both components in c_1^r and c_2^{2t} for $1 \leq r \leq k!$ and $1 \leq t \leq k!/(k-1)$ maximizes $|\sigma(\{c\}) \cap U|$ to $k - 1$ and again the algorithm may select the components in the second group. Similarly, in the next $k!/(k-i+1)$ iterations for $i = k-2 \ldots 1$, both components in c_1^r and c_2^{it} for $1 \leq r \leq k!$ and $1 \leq t \leq k!/(k-i+1)$ maximizes $|\sigma(\{c\}) \cap U|$ to $k-i+1$. Therefore after $k! \cdot H(k)$ iterations, the algorithm may select all the components of the form c_2^{st}, while the optimum solution is the components of the form c_1^r. The ratio of the size of the greedy solution to the optimum solution is $k! \cdot H(k)/k! = H(k)$ which is equal to the bound proved in Theorem 3.

3. Component Selection with Real Costs

In this section, we extend CSUC by assigning a positive real number to each component or composition as its cost; therefore we call this extension Component Selection with Real Costs (CSRC). This problem has more applications than the previous

one. For example, suppose that a few components satisfy most of the objectives of the problem, but they need many modification and integration efforts. Moreover, we can use a lot of cheaper components to satisfy the same objectives. The question here is: should we select either few costly components or many inexpensive components? This problem can easily be modeled by CSRC.

The formal definition of the decision version of CSRC is as follows:

CSRC

Input: A set $C = \{c_1, c_2, \ldots, c_m\}$ of components, a set $O = \{o_1, o_2, \ldots, o_n\}$ of objectives, an oracle function $\sigma : 2^C \to 2^O$, a cost function $\omega : 2^C \to \mathbb{R}^+$, and a positive real number $K \leq |C|$.

Question: Is there a composition $S \subseteq C$ such that $\omega(S) \leq K$ and $O \subseteq \sigma(S)$?

CSRC is as hard as CSUC, since it contains CSUC as a special case. When the cost of each composition is equal to the number of components it contains, i.e. $\omega(S) = |S|$, the problem converts to CSUC. Therefore we immediately conclude the following theorem.

Theorem 4. *CSRC is NP-complete.*

Because CSRC is NP-complete, we hope to find an approximation algorithm for CSRC as we did for CSUC. Since CSUC is a special case of CSRC, similar results about the complexity of CSRC can be obtained. Specifically, CSRC cannot be approximated to within $o(\sqrt{m})$ if the number of emergent compositions is unbounded, and cannot be approximated at all if the number of anti-emergent compositions is unbounded. In the case of non-emergent compositions, like CSUC, it cannot be approximated to within $(1 - o(1)) \cdot \ln n$ unless P=NP. This is because it is as hard as Minimum Set Cover, which cannot be approximated to within $(1-o(1)) \cdot \ln n$ unless P = NP [6]. Therefore, we should not expect a better algorithm than the greedy algorithm.

CSRC has an additional function, ω, that can exhibit three different behaviors like σ. These behaviors are *cumulative*, *convergent* and *divergent*. When the cost of a composition is equivalent to the sum of the costs of its components the composition is cumulative. When the cost is less than that value, the composition is convergent and when it is greater, the composition is divergent.

It can be proved, as was the case for emergent and anti-emergent behaviors, that when CSRC has unbounded number of convergent compositions, it cannot be approximated to within $o(\sqrt{m})$, and when has unbounded number of divergent compositions, it cannot be approximated at all. In contrast to these behaviors, when CSRC contains only cumulative compositions, it can be approximated to within $O(\log n)$. The approximation algorithm is a modified version of the greedy algorithm for CSUC and is shown in Algorithm 2.

Theorem 5. Greedy-CSRC *when σ is non-emergent and ω is cumulative, approximates the solution to within $H(k) = \sum_{i=1}^{k} \frac{1}{i}$ where $k = \max\{|\sigma(\{c\})| : c \in C\}$.*

Algorithm 2. The algorithm for CSRC with non-emergent and cumulative behavior.

```
1: function GREEDY-CSRC(C, O, σ, ω)
2:     U ← O                                   ▷ To be satisfied objectives
3:     S ← ∅                                   ▷ Selected components
4:     while U ≠ ∅ do
5:         Find the most cost effective component,
               i.e. c_i ∈ C that minimizes ω(c_i) / |σ({c_i}) ∩ U|
6:         S ← S ∪ {c_i}                       ▷ Select component
7:         U ← U − σ({c_i})                    ▷ Remove satisfied objectives
8:         return S
9:     end while
10: end function
```

Proof. Imagine that when the greedy algorithm chooses a component c, it evenly divides the cost of choosing c between any objective newly satisfied by c. At the end, the total cost of satisfying all the objectives equals the cost of the greedy solution.

Consider a component c selected in the optimal solution, and assume $\sigma(\{c\}) = \{o'_1, o'_2, \ldots, o'_l\}$, and the greedy algorithm satisfies these objectives in the order $o'_l, o'_{l-1}, \ldots, o'_1$. When the greedy algorithm satisfies o'_j, for $1 \leq j \leq l$, at least j objectives of c are still unsatisfied, and because the greedy selection strategy is to choose the most cost effective component, it charges o'_j at most $\omega(c)/j$, because c at this iteration is a possible choice. Summing this value over all j, the total cost paid for all objectives satisfied by c is at most $\omega(c) \cdot H(l)$, and summing over all components selected in the optimum solution, the overall cost is at most $\mathrm{OPT} \cdot H(k)$, where k is the maximum number of objectives satisfied by a component and OPT is the cost of the optimum solution.

In order to show that the ratio is tight, consider the instance in which we have n objectives and $n+1$ components. Each component c_i, for $1 \leq i \leq n$, has cost $\omega(c_i) = 1/(n-i+1)$ and satisfies o_i. The component c_{n+1} has cost $\omega(c_{n+1}) = 1+\epsilon$, for some small $\epsilon > 0$, and satisfies all the objectives. Clearly the optimum composition consists of only c_{n+1} with cost $1+\epsilon$, but the greedy algorithm selects the components c_i for $1 \leq i \leq n$, because in the i-th iteration of the **while** loop of Algorithm 2 for $1 \leq i \leq n$, the most cost effective component is c_i with effective cost $\frac{1}{n-i+1}$ while the component c_{n+1} has effective cost $\frac{1+\epsilon}{n-i+1}$. Therefore in this iteration, the component c_i will be selected, and after n iterations all the objectives will be satisfied. The total cost in this solution is $\sum_{i=1}^{n} 1/(n-i+1) = H(n)$, while the cost of the optimum solution is $1+\epsilon$. Thus the ratio will be $H(n)/(1+\epsilon)$ and with a very small ϵ, the upper bound will be obtained. □

4. Component Selection with Multi-Costs

Although CSRC in its general form is strong enough to model all situations, its inapproximability when the cost function is convergent or divergent makes CSRC less usable. In this section we extend the component selection problem to alleviate this shortcoming.

This problem extends *cumulative* CSRC by adding a more general cost function to components; therefore we call the problem Component Selection with Multi-Costs (CSMC). In CSMC we have a set of known costs, and assign to each component a subset of these costs. The goal is to choose a composition such that the union of the costs charged by the selected components has the smallest weight.

As a simple example, assume there are two component providers, A and B, each providing two components to satisfy objectives o_1 and o_2. Company A provides c_{A1} and c_{A2}, and B provides c_{B1} and c_{B2}. We know c_{A1} and c_{B1} satisfy o_1, while c_{A2} and c_{B2} satisfy o_2. Furthermore, A sells its components in a package with cost \$1.5, whereas the cost of each component provided by B is \$1. It means that if we use any or both components of A, we should pay \$1.5. If we model this problem with CSRC, the cost function $\omega(\cdot)$ will be convergent, and therefore Algorithm 2 is not applicable. On the other hand, if we model the problem with CSMC, as we will see, there is an approximation algorithm for it.

The formal definition of the decision version of CSMC is as follows:

CSMC

Input: A set $C = \{c_1, c_2, \ldots, c_m\}$ of components, a set $O = \{o_1, o_2, \ldots, o_n\}$ of objectives, a set $D = \{d_1, d_2, \ldots, d_p\}$ of costs, an oracle function $\sigma : 2^C \to 2^O$, a cost function $\omega : C \to 2^D$, a weight function for costs $\delta : D \to \mathbb{R}^+$, and a positive real number K.

Question: Is there a composition $S \subseteq C$ such that $\sum_{d \in \bigcup_{c \in S} \omega(c)} \delta(d) \leq K$ and $O \subseteq \sigma(S)$?

In CSRC, for each composition S, $\omega(S)$ will return a real number specifying the cost of using S. On ther other hand, ω in CSMC is defined over the set of components, and for each component c, $\omega(c)$ will return a subset of costs in D imposed by using c in the composition. The cost of a composition is the union of all costs charged by the components of that composition. The goal in the problem is to minimize the total weights of costs for the composition.

This problem has two differences with CSRC. First, in CSRC we have only one type of cost, which is a real number for each composition, whereas in CSMC we may have several types of costs. Second, some components in CSMC may have common costs. This means if we select a component, we can use other components with the same cost without paying the cost twice.

The general form of CSRC is more powerful than the general form of CSMC, so we may not see an immediate benefit of studying CSMC. If we are to use a convergent cost function in CSRC, we can convert any instance of CSMC to an instance

of CSRC. However, in the following, we show that an approximation algorithm exists for CSMC. This makes CSMC look very different from CSRC. In fact, CSMC extends cumulative CSRC, and makes it possible to have convergent compositions while being approximable.

In terms of complexity, CSMC is at least as hard as CSRC with cumulative behavior. To prove this claim, we should convert each instance of CSRC with cumulative behavior to CSMC. Let $P = (C, O, \sigma, \omega)$ be an instance of CSRC. We create $P' = (C', O', D', \sigma', \omega', \delta')$ an instance of CSMC. Let $C' = C$ and $O' = O$ and $\sigma' = \sigma$. For each component, $c_i \in C$, add a cost d_{c_i} to D' and let $\omega'(c_i) = d_{c_i}$ and $\delta'(d_{c_i}) = \omega(c_i)$. Now, for each composition $S \subseteq C$ of components, the set of objectives satisfied for P is equal to the set of objectives satisfied for P'. Similarly, the cost of compositions for P is equal to the cost of compositions for P'. Therefore the two problems are equivalent, and we can conclude the following theorem.

Theorem 6. *CSMC cannot be approximated to within* $(1 - o(1)) \cdot \ln n$, *where* $n = |O|$, *unless* $P = NP$.

We can prove a stronger claim about the approximation ratio of CSMC. This ratio comes from another problem called *Red Blue Set Cover*. The definition of the decision version of the problem is as follows:

Red Blue Set Cover

Input: Two finite sets $R = \{r_1, r_2, \ldots, r_\rho\}$ and $B = \{b_1, b_2, \ldots, b_\beta\}$, a family $\mathcal{S} \subseteq 2^{R \cup B}$ of n subsets of $R \cup B$, and an integer $K \leq |S|$.

Question: Is there a subfamily $\mathcal{C} \subseteq \mathcal{S}$ that covers all elements of B, but covers at most K elements of R?

This problem is shown to be NP-complete [3]. It has been proved that the problem cannot be approximated to within $2^{log^{1-\epsilon} n}$, for any $0 < \epsilon < 1$, under some plausible complexity theoretic assumptions, such as P $\neq$ NP or NP $\nsubseteq$ DTIME($n^{O(\text{polylog } n)}$) [14]. In the following, we show that CSMC when σ has only non-emergent behavior is as hard as Red-Blue Set Cover (RBSC) and hence a similar lower bound on its approximation ratio arises.

Theorem 7. *Non-emergent CSMC cannot be approximated to within* $2^{log^{1-\epsilon} n}$, *unless* $P = NP$.

Proof. The proof is done by presenting an approximation ratio preserving reduction from RBSC to CSMC. Let $P = (B, R, \mathcal{S})$ be an instance of RBSC. We will create $P' = (C', O', D', \sigma', \omega', \delta')$, an instance of non-emergent CSMC, such that P has a solution if and only if P' has a solution. For any $b_i \in B$, $1 \leq i \leq \beta$, add an objective o_i to O'. For any $r_j \in R$, $1 \leq j \leq \rho$, add a cost d_j to D'. For any $S_l \in \mathcal{S}$, $1 \leq l \leq n$, add a component c_l to C'. For any $b_i \in S_l$, add o_i to $\sigma(\{c_l\})$ and for any $r_j \in S_l$, add d_j to $\omega(c_l)$. For any $d_j \in D'$ let $\delta(d_j) = 1$. We assume the set of objectives satisfied by a composition is equal to the union of the set of objectives

satisfied by each component in the composition. Obviously, this creates an instance of non-emergent CSMC.

Now it is sufficient to show that: (i) any subfamily $\mathcal{C} \subseteq \mathcal{S}$ that covers all elements of B has an equivalent composition $S' \subseteq C'$ that satisfies all objectives in O' and the number of elements of R covered by $\mathcal{C}$ is equal to the number of elements of D' imposed by S', that is the cost of S', and (ii) any composition $S' \subseteq C'$ with the specified requirements has an equivalent subfamily in $\mathcal{C} \subseteq \mathcal{S}$, with equal cost.

We here prove the second claim. The other claim is proved similarly. Let S' be a composition that satisfies all objectives in O'. According to the construction of C' and σ, there must be a subcollection $\mathcal{C} \subseteq \mathcal{S}$ such that $B \subseteq \bigcup_{S \in \mathcal{C}} S$. We also have

$$\begin{aligned}
\mathrm{Cost}(S') &= \sum_{d_i \in \bigcup_{c_l \in S'} \omega(c_l)} \delta(d_i) \\
&= \sum_{d_i \in \bigcup_{c_l \in S'} \omega(c_l)} 1 \\
&= \left| \bigcup_{c_l \in S'} \omega(c_l) \right| \\
&= \left| \bigcup_{S_l \in \mathcal{C}} (R \cap S_l) \right| \\
&= \text{The number of elements of } R \text{ covered by } \mathcal{C}.
\end{aligned}$$

Therefore the cost of S' is equal to the cost of $\mathcal{C}$.

From the above arguments, it follows that any approximation algorithm for CSMC is an approximation algorithm for RBSC with the same approximation ratio. Since RBSC cannot be approximated to within $2^{log^{1-\epsilon} n}$ for any $0 < \epsilon < 1$, CSMC cannot be approximated to within the same bound. □

In spite of the lower bound for the approximation ratio of RBSC, the best known approximation algorithm has the ratio $2\sqrt{n \log \beta}$ where $n = |\mathcal{S}|$ and $\beta = |B|$. This algorithm was introduced by Peleg [14]. In Algorithm 3, we present an adaptation of Peleg's greedy algorithm for CSMC.

Theorem 8. Approx-CSMC *approximates CSMC to within* $2\sqrt{m \log p}$, *where* m *is the number of components and* p *is the number of different costs.*

Proof. The approximation ratio of the proposed algorithm by Peleg (Low-Edge2 [14]) is $2\sqrt{|\mathcal{S}| \log |B|}$, which means the ratio between the cost of the approximate solution with the optimum is at most $2\sqrt{|\mathcal{S}| \log |B|}$. Algorithm 3 is a direct adaptation of Peleg's algorithm for CSMC, based on the transformation used in the proof of Theorem 7. Therefore, the same approximation ratio, that is $2\sqrt{m \log p}$, holds for CSMC, where m is the number of components and p is the number of costs. □

Algorithm 3. The approximation algorithm for CSMC problem.

function GREEDY-CSMC$(C, O, D, \sigma, \omega, \delta)$
  Modify CSMC instance into an instance $\mathcal{T}$ of CSRC as follows:
    $C_{\mathcal{T}} \leftarrow C, \quad O_{\mathcal{T}} \leftarrow O, \quad \sigma_{\mathcal{T}} \leftarrow \sigma,$
    $\omega_{\mathcal{T}} : 2^C \rightarrow \mathbb{R}^+$ such that $\forall c_i \in C$, $\omega_{\mathcal{T}}(c_i) \leftarrow \sum_{d_j \in \omega(c_i)} \delta(d_j)$.
  return GREEDY-CSRC$(C_{\mathcal{T}}, O_{\mathcal{T}}, \sigma_{\mathcal{T}}, \omega_{\mathcal{T}})$.
end function

function BAD-COSTS$(C, O, D, \sigma, \omega, \delta, X)$
  $C_X \leftarrow \{c_i \in C : \sum_{d_j \in \omega(c_i)} \delta(d_j) \leq X\}$. ▷ Discard components with cost more than X
  If $O \nsubseteq \sigma(C_X)$ **return** C. ▷ C_X is not feasible
  $n \leftarrow |C|$, $\beta \leftarrow |O|$.
  $Y \leftarrow \sqrt{\frac{n}{\log \beta}}$.
  Separate costs to *good* and *bad* costs:
    $D_G \leftarrow \{d_j \in D : \sum_{c_i : d_j \in \omega(c_i)} 1 > Y\}$
    $D_B \leftarrow D - D_G$.
  Discard the costs in D_G and define $\omega_{X,Y} : 2^{C_X} \rightarrow 2^{D_B}$
    such that for all $c_i \in C_X$, $\omega_{X,Y}(c_i) \leftarrow \omega(c_i) - D_G$.
  return GREEDY-CSMC$(C_X, O, D_B, \sigma, \omega_{X,Y}, \delta)$.
end function

function APPROX-CSMC$(C, O, D, \sigma, \omega, \delta))$
  $t \leftarrow \sum_{d_j \in D} \delta(d_j)$.
  $S_{\min} \leftarrow C$.
  for $X \leftarrow 1$ to t **do**
    $S \leftarrow$ BAD-COSTS$(C, O, D, \sigma, \omega, \delta, X)$.
    if Cost$(S_{\min})$ > Cost(S) **then**
      $S_{\min} \leftarrow S$.
    end if
  end for
  return $S_{\min}$.
end function

5. Conclusion

Composability is the ability to combine reusable simulation components to satisfy a set of user objectives. Composability is a highly demanded goal for model and simulation developers because of the benefits afforded by reuse. Component Selection (CS) is an NP-complete optimization problem formally shown to be embedded within composability.

The first important result of this paper was that CS in the general formulation is not approximable. We proved this claim in two cases: compositions have emergent rules and compositions have anti-emergent rules. We showed in the first case the problem is not approximable to within $o(\sqrt{m})$, and in the second case it is not approximable to within any ratio.

In the next part of the paper, we considered three special versions of CS. In the first problem each component has a unit cost. In the second one, each component has a real number as its cost. The third problem is the case that each component can have several types of costs with different weights, and also each component may have some costs common with other components. For each problem, we first formally defined the problem, and then gave an approximation algorithm to solve it.

References

[1] P. Baker, M. Harman, K. Steinhofel and A. Skaliotis, Search based approaches to component selection and prioritization for the next release problem, in *Proceedings of the 22nd IEEE International Conference on Software Maintenance*, *ICSM '06* (IEEE Computer Society, Washington, DC, USA, 2006), pp. 176–185.

[2] C. Berger, M. Chaudron, R. Heldal, O. Landsiedel and E. M. Schiller, Model-based, composable simulation for the development of autonomous miniature vehicles, in *Proceedings of the Symposium on Theory of Modeling & Simulation — DEVS Integrative M&S Symposium*, *DEVS 13* (2013), pp. 17:1–17:8.

[3] R. D. Carr, S. Doddi, G. Konjevod and M. Marathe, On the red-blue set cover problem, in *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, *SODA '00* (2000), pp. 345–353.

[4] C.-K. Chui, Z. Wang, J. Zhang, J. S.-K. Ong, L. Bian, J. C.-M. Teo, C.-H. Yan, S.-H. Ong, S.-C. Wang, H.-K. Wong and S.-H. Teoh, A component-oriented software toolkit for patient-specific finite element model generation, *Advances in Engineering Software* **40** (2009) 184–192.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, 3rd edn. (The MIT Press, 2009).

[6] I. Dinur and D. Steurer, Analytical approach to parallel repetition, in *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, *STOC '14* (ACM, New York, NY, USA, 2014), pp. 624–633.

[7] M. Dolenc, Developing extendible component-oriented finite element software, *Advances in Engineering Software* **35** (2004) 703–714.

[8] S. A. Fahmi and H.-J. Choi, A study on software component selection methods, in *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on* **01** (2009), pp. 288–292.

[9] M. R. Fox, D. C. Brogan and P. F. Reynolds, Jr., Approximating component selection, in *Proceedings of the 36th Conference on Winter Simulation*, *WSC '04* (Winter Simulation Conference, 2004), pp. 429–434.

[10] D. S. Johnson, Approximation algorithms for combinatorial problems, *Journal of Computer and System Sciences* **9**(3) (1974) 256–278.

[11] M. A. Khan and S. Mahmood, A graph based requirements clustering approach for component selection, *Advances in Engineering Software* **54** (2012) 1–16.

[12] M. Nouri and J. Habibi, Approximating component selection with general costs, *Proceeding of 13th International CSI Computer Conference* (Springer, 2008), pp. 61–68.

[13] E. H. Page and J. M. Opper, Observations on the complexity of composable simulation, in *Proceedings of the 36th Conference on Winter Simulation Winter Simulation Conference*, *WSC '04* (Winter Simulation Conference, 1999), pp. 553–560, http://www.cs.virginia.edu/˜rgb2u/03F-SIW-072.pdf.

[14] D. Peleg, Approximation algorithms for the Label-Cover$_{\mathrm{MAX}}$ and red-blue set cover problems, *J. Discrete Algorithms* **5**(1) (2007) 55–64.

[15] M. D. Petty, E. W. Weisel and R. R. Mielke, Computational complexity of selecting components for composition, *Fall 2003 Simulation Interoperability Workshop* (2003).

[16] E. G. Roselló, M. J. Lado, A. J. Méndez, J. G. Dacosta and M. P. Cota, A component framework for reusing a proprietary computer-aided engineering environment, *Advances in Engineering Software* **38** (2007) 256–266.

[17] H. Wang, S. X.-D. Tan, D. Li, A. Gupta and Y. Yuan, Composable thermal modeling and simulation for architecture-level thermal designs of multicore microprocessors, *ACM Transactions on Design Automation of Electronic Systems* **18** (2013) 28:1–28:27.