Approximating Component Selection with General Costs

Mostafa Nouri and Jafar Habibi

Computer Engineering Department,
Sharif University of Technology, Tehran, Iran
nourybay@ce.sharif.edu, jhabibi@sharif.edu

Abstract. 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 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 second 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 examine approximating solutions that make the CS 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 variations of Component Selection, we prove that the general Component Selection problem is inapproximable.

Keywords: Component Selection, NP-Completeness, Set Cover, Approximation Algorithms, Red-Blue Set Cover.

1 Introduction

A recent topic of interest in the software engineering community and the modeling and simulation community involves the notion of component-based software development. There had been many efforts in the software community to incorporate the concept of reusing. Components offer a useful mechanism to support reuse. But a number of questions are raised by them as well.

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 the problem of choosing the minimum number of components from a set of components such that their composition satisfies a set of objectives. CS which is an NP-complete [5] optimization problem formally shown to be embedded within composability [5]. For composability to be achievable in reasonable time, the selection of a set of components must be accomplishable in polynomial-time. Therefore we should try to find a polynomial-time algorithm

H. Sarbazi-Azad et al. (Eds.): CSICC 2008, CCIS 6, pp. 61–68, 2008.

to solve CS Problem. In [2] it has been conjectured that in the general case CS is inapproximable.

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

- Prove that Component Selection problem is inapproximable in the general case, a conjecture which is suggested by Fox *et al.* in [2].
- Give an approximation algorithm for Component Selection problem, when components have unit costs and compositions have not unbounded emergent and non-emergent behaviors. We also give the upper bound of the approximation ratio.
- Define two new version of Component Selection problem, with real or general costs, and offer the approximation algorithms and the upper bound of the approximation ratios.

In the following we will have examine component selection in three different cases. In section 2 we will study the well-known problem of component selection with unit cost. Then in section 3 we will extend the problem to the case that components have some real valued costs. In section 4 the general component selection are defined and the approximation algorithm for it is presented. Finally in section 5 we will summarize the results of this paper.

2 Component Selection with Unit Costs

In this section we consider the simplest form of component selection, CSUC, in which adding each component poses a unit cost to the composition. Informally the problem is to select and compose minimum number of components from a repository of components such that the composition will meet a given set of objectives.

Let $O = \{o_1, o_2, ..., o_n\}$ be a set of objectives and $C = \{c_1, c_2, ..., c_m\}$ 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 composition of components, e.g. $(c \circ c')$ is the composition of c and c'. Let $\vDash$ and $\nvDash$ denote *satisfying* and *not satisfying* an objective respectively, e.g. $c \vDash o$ means component c satisfies objective o and $c \nvDash o$ means component c does not satisfy objective 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 satisfied objectives 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 [3] and the third one (anti-emergent) was later defined by Petty *et al.* [5]. Informally if none of c and c' and $c \circ c'$ satisfy o, then the composition is *non-emergent*. If c

Table 1. Different types of compositions

Objective Satisfaction of Components		Objective Satisfaction of Composition	Composition Type
$c \models o$	$c' \models o$	$(c \circ c') \models o$	Non-emergent
$c \not\models o$	$c' \models o$	$(c \circ c') \models o$	Non-emergent
$c \models o$	$c' \not\models o$	$(c \circ c') \models o$	Non-emergent
$c \not\models o$	$c' \not\models o$	$(c \circ c') \not\models o$	Non-emergent
$c \not\models o$	$c' \not\models o$	$(c \circ c') \models o$	Emergent
$c \models o$	$c' \models o$	$(c \circ c') \not\models o$	Anti-emergent
$c \not\models o$	$c' \models o$	$(c \circ c') \not\models o$	Anti-emergent
$c \models o$	$c' \not\models o$	$(c \circ c') \not\models o$	Anti-emergent

and c' do not satisfy o but $c \circ c'$ satisfy 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.* [5] 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:

> *Instance*: Set $C = \{c_1, c_2, ..., c_m\}$ of components, set $O = \{o_1, o_2, ..., o_n\}$ of objectives, oracle function $\sigma : 2^C \rightarrow 2^O$, positive integer $K \leq |C|$.
> *Question*: Does C contain a composition S that satisfies O of size K or less, i.e. a subset $S \subseteq C$ with $|S| \leq K$ such that $O \subseteq \sigma(S)$?

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

Fox *et al.* [2] have demonstrated that the greedy algorithm is not an approximation algorithm for CSUC. Since the greedy algorithm is one of the best algorithm for Minimum Set Cover problem which is related to the CSUC, they conjectured that there is no algorithm that can approximate the CSUC problem. We here claim that in the general case, the CSUC problem cannot be approximated.

Theorem 1. *There is no approximation algorithm for CSUC when the compositions in CSUC may have some emergent or anti-emergent rules.*

Proof. The proof is in the full version of the paper.

While the general CSUC problem is not approximable, we can use the greedy algorithm as it is used for Minimum Set Cover problem and get an approximation ratio of $H(k) = \sum_{i=1}^{k} \frac{1}{i}$ where $k = \max\{|\sigma(c)| : c \in C\}$, when the composition only exhibits non-emergent behavior. Also when the number of emergent and anti-emergent behaviors are polynomially bounded and can be iterated by σ, we can use a modified version of the greedy algorithm and get the same ratio. The greedy algorithm when CSUC only exhibits non-emergent behavior can be seen in Algorithm 1. In the following theorem it will be proved that the approximation ratio of the algorithm is $H(k)$.

Algorithm 1. The algorithm for CSUC problem 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                         ▷ Some unsatisfied objective
5:         Select a c_i ∈ C that maximizes |σ(c_i) ∩ U|
6:         U ← U − σ(c_i)                     ▷ Remove satisfied objectives
7:         S ← S ∪ {c_i}                      ▷ Select component
8:         return S
9:     end while
10: end function
```

Theorem 2. *The algorithm* 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. The proof is in the full version of the paper.

3 Component Selection with Real Costs

In this section we extend the previous problem of component selection by assigning a real number as a cost to each component or composition, therefore we call it Component Selection with Real Costs (CSRC). This problem has more application 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. But we can also use some more cheap components to satisfy the same objectives. The problem in this case is "what should we do?". Should we select few costly components or many inexpensive components? This problem can easily been modeled by CSRC. The formal definition of the decision version of CSRC is as follows:

> *Instance*: Set $C = \{c_1, c_2, ..., c_m\}$ of components, set $O = \{o_1, o_2, ..., o_n\}$ of objectives, oracle function $\sigma : 2^C \to 2^O$, cost function $\omega : 2^C \to \mathbb{R}^+$, positive integer $K \leq |C|$.
> *Question*: Does C contain a composition S that satisfies O and its cost is K or less, i.e. a subset $S \subseteq C$ such that $\omega(S) \leq K$ and $O \subseteq \sigma(S)$?

CSRC is as hard as the CSUC, since it contains the 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 changes into CSUC. So we can immediately get the following theorem.

Theorem 3. *Component selection with real costs (CSRC) is NP-complete.*

As the previous problem, because the problem is NP-complete we should seek an approximation algorithm for CSRC. CSRC is also at least as hard as CSUC, and CSUC cannot be approximated with ratio better than $O(\log n)$ unless P=NP (since it is as hard as minimum set cover problem), so we should not expect a better algorithm than CSUC.

CSRC problem add a new function to CSUC. This function, like σ, can exhibits three different behaviors. These behaviors are *cumulative*, *convergent* and *divergent*. The descriptions are easy, when the cost of a composition is equivalent to the sum of the costs of its components the composition is cumulative. When its cost is less than that value, the composition is convergent and when it is greater, the composition is divergent.

It can be proved, like emergent and anti-emergent behaviors, that when CSRC problem has unbounded number of convergent or divergent compositions, it cannot be approximated. In contrast to these behaviors, when CSRC contains only cumulative composition, it can be approximated with a similar ratio. The algorithm is a modified version of greedy algorithm on the CSUC. The algorithm can be seen in the Algorithm 2.

Algorithm 2. The algorithm for CSRC problem 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                                 ▷ Some unsatisfied objective
5:         Find the most cost effective component,
                   i.e. c_i ∈ C that minimizes ω(c_i) / |σ(c_i) ∩ U|
6:         U ← U − σ(c_i)                             ▷ Remove satisfied objectives
7:         S ← S ∪ {c_i}                              ▷ Select component
8:         return S
9:     end while
10: end function
```

Theorem 4. *The* GREEDY-CSRC *when σ has no emergent or anti-emergent and ω has no convergent or divergent behavior, approximates the solution with ratio $H(n) = \sum_{i=1}^{n} \frac{1}{i}$ where $n = |O|$.*

Proof. The proof is in the full version of the paper.

4 Component Selection with Multi-costs

In this section we extend the component selection problem further. The problem is extended by adding a more general cost to components and compositions. Therefore we call it Component Selection with Multi-Costs (CSMC). In this problem we define some known costs, and assign to each components and/or compositions a subset of these costs. The problem in this case is to make a composition such that the minimum number of these costs are selected. The formal definition of the decision version of this problem is as follows:

> *Instance*: Set $C = \{c_1, c_2, ..., c_m\}$ of components, set $O = \{o_1, o_2, ..., o_n\}$ of objectives, set $D = \{d_1, d_2, ..., d_p\}$ of costs, oracle function $\sigma : 2^C \rightarrow 2^O$, cost function $\omega : 2^C \rightarrow 2^D$, weight function for costs $wd : D \rightarrow \mathbb{R}^+$, positive integer $K \leq |C|$.
> *Question*: Does C contain a composition S that satisfies O and the total weight of its costs is K or less, i.e. a subset $S \subseteq C$ such that $\sum_{d \in \omega(S)} wd(d) \leq K$ and $O \subseteq \sigma(S)$?

In this problem, ω has been changed. In CSRC, ω will return a real number, specifying the cost of using that composition. But in CSMC, ω will return a subset of costs in D and the goal is to minimize the total weight of cost set for the total composition. This problem has two difference with CSRC. First, the compositions in CSRC has only one type of cost, which can be specified by a real number, whereas in CSMC components/compositions may have several types of costs. Second, some components/compositions in CSMC may have common costs, which means if we select a component/composition, we can select the other components/compositions without paying that cost twice. Consequently CSMC is a more powerful problem and can be used to model more complicated cases.

In view of complexities of the two problems, CSMC is at least as hard as *approximable* CSRC. To prove this claim we should convert each instance of CSRC to CSMC problem. Let $P = (C, O, \sigma, \omega)$ be an instance of CSRC. We need to create $P' = (C', O', D', \sigma', \omega', wd')$ 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 assign it to $\omega'(c_i)$ and set its weight $wd(d_{c_i})$ to $\omega(c_i)$. If ω has convergent or divergent behavior, and the convergent and divergent compositions are polynomially bounded and can be iterated by σ (P is approximable), add corresponding compositions to C' and new costs to D' and assign these costs to the corresponding compositions likewise. Now if P' has an optimum solution with cost w, then there is a $S' \subseteq C'$ such that $O' \subseteq \sigma(S')$ and $\sum_{d \in \omega(S')} wd(d) = w$, for the corresponding subset S' in the problem P, also $O \subseteq \sigma(S)$ (since C, O and σ are equal in both problems) and $\omega(S) = w$ (since the costs of components S' are equivalent to the costs of components of S. Therefore we can conclude the following theorem.

Theorem 5. *CSMC problem cannot be approximated by a ratio better than* $H(n) = \sum_{i=1}^{n} \frac{1}{i}$ *where* $n = |O|$.

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

> *Instance*: Let $R = \{r_1, r_2, ..., r_\rho\}$ and $B = \{b_1, b_2, ..., b_\beta\}$ be two disjoint finite sets, and let $\mathcal{S} \subseteq 2^{R \cup B}$ be a family of subsets of $R \cup B$, and let $K \leq |S|$ be an integer.
> *Question*: Does $\mathcal{S}$ contains a subfamily $\mathcal{C} \subseteq \mathcal{S}$ that covers all elements of B, but covers at most K elements of R.

Theorem 6. *CSMC is as hard as RBSC problem.*

Proof. The proof is in the full version of the paper.

Algorithm 3. The approximation algorithm for CSMC problem.

function GREEDY-CSMC($C, O, D, \sigma, \omega, wd$)
  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 $\forall c_i \in C$, $\omega_\mathcal{T}(c_i) \leftarrow \sum_{d_j \in \omega(c_i)} wd(d_j)$.
  return GREEDY-CSRC($C_\mathcal{T}, O_\mathcal{T}, \sigma_\mathcal{T}, \omega_\mathcal{T}$).
end function
function BAD-COSTS($C, O, D, \sigma, \omega, wd, X$)
  Discard components with total cost more than X, i.e.
    $C_X \leftarrow \{c_i \in C | \sum_{d_j \in \omega(c_i)} wd(d_j) \leq X\}$.
  If $O \nsubseteq \sigma(C_X)$ **return** C. ▷ C_X is not feasible
  $Y \leftarrow \sqrt{\frac{n}{\log \beta}}$.
  Separate the costs to *good* and *bad* costs:
    $D_G \leftarrow \{d_j \in D | \sum_{c_i : d_j \in \omega(c_i)} 1 > Y\}, \quad D_B \leftarrow D - D_G$.
  Discard the costs in D_G and set
    $\omega_{X,Y} = 2^C \rightarrow 2^D_G$ such that for all $c_i \in C_X$, $\omega_{X,Y}(c_i) \leftarrow \omega(c_i) - D_G$.
  return GREEDY-CSMC($C_X, O, D_B, \sigma, \omega_{X,Y}, wd$).
end function
function APPROX-CSMC($C, O, D, \sigma, \omega, wd$))
  for $X \leftarrow 1$ to ρ **do**
    call BAD-COSTS($C, O, D, \sigma, \omega, wd, X$).
    Compute the weight of selecting the returned components.
  end for
  return the solution with the least cost.
end function

This problem has been proved to be NP-complete [1]. It has also been proved that this problem cannot be approximated with ratio $2^{log^{1-\epsilon} n}$ for any $0 < \epsilon < 1$ under some plausible complexity theoretic assumptions (such as P $\neq$ NP or NP $\nsubseteq$ DTIME($n^{O(\text{polylog } n)}$). We here show that CSMC is as hard as Red-Blue Set Cover (RBSC) and hence the same bound on its approximation ratio arises.

In spite of the lower bound for approximation ratio of RBSC problem, the best known algorithm for approximating the solution has the ratio $2\sqrt{n \log \beta}$

where $n = |\mathcal{S}|$ and $\beta = |B|$. This algorithm was introduced by Peleg [4]. We can use his algorithm for CSMC problem by transforming each instance of CSMC into an instance of RBSC and then applying the algorithm on this problem. After solving RBSC problem, we can transform the solution backward to find the approximate solution of CSMC. In this section, instead we modify Peleg' algorithm [4] and make it suitable for CSMC problem. The details of greedy algorithm for CSMC can be seen in Algorithm 3.

Theorem 7. *The approximation ratio of the found solution with the optimum solution of CSMC problem is at most* $2\sqrt{n \log \beta}$.

Proof. The proof is in the full version of the paper.

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 main result of this paper is that CS in the general formulation is not approximable. Another result of the paper is the definition of three modified versions of CS along with their approximation algorithms. In the first problem each component has a unit cost. In the second problem, each component has a real number as its cost and in the third (and the most general) problem each component may have several types of costs with different weights, and also each component may have some costs common with other components.

For future works we want to seek algorithms with better approximation ratios for these problems (specially for CSMC problem, which there is a gap between the proved upper bound and lower bound). Also we will study other modified versions of Component selection which are tractable by polynomial algorithms.

References

1. Carr, R.D., Doddi, S., Konjevod, G., Marathe, M.V.: On the red-blue set cover problem. In: SODA, pp. 345–353 (2000)
2. Fox, M.R., Brogan, D.C., Reynolde Jr., P.F.: Approximating component selection. In: Winter Simulation Conference, pp. 429–434 (2004)
3. Page, E.H., Opper, J.M.: Observations on the complexity of composable simulation. In: Winter Simulation Conference, pp. 553–560 (1999)
4. Peleg, D.: Approximation algorithms for the label-cover$_{\max}$ and red-blue set cover problems. J. Discrete Algorithms 5(1), 55–64 (2007)
5. Petty, M.D., Weisel, E.W., Mielke, R.R.: Computational complexity of selecting components for composition. In: Fall 2003 Simulation Interoperability Workshop, pp. 553–560 (2003)