Efficient Placement of Multi-Component Applications in Edge Computing Systems

Tayebeh Bahreini (*Ph.D. Student*)
Dept. of Computer Science
Wayne State University
Detroit, Michigan 48202
tayebeh.bahreini@wayne.edu

Daniel Grosu
Dept. of Computer Science
Wayne State University
Detroit, Michigan 48202
dgrosu@wayne.edu

ABSTRACT

One of the main challenges in Mobile Edge Computing (MEC) is determining an efficient placement of the components of a mobile application on the edge servers that minimizes the cost incurred when running the application. We address the problem of multi-component application placement in edge computing by designing an efficient on-line algorithm that solves it. We also introduce a Mixed Integer Linear Programming formulation of the multi-component application placement problem that takes into account the dynamic nature of users' location and the network capabilities. We perform extensive experiments to evaluate the performance of the proposed algorithm. Experimental results indicate that the proposed algorithm has very small execution time and obtains near optimal solutions.

CCS CONCEPTS

•Networks → Cloud computing; •Computer systems organization → Cloud computing;

KEYWORDS

edge computing, services, component placement algorithm

ACM Reference format:
Tayebeh Bahreini (*Ph.D. Student*) and Daniel Grosu. 2017. Efficient Placement of Multi-Component Applications in Edge Computing Systems. In *Proceedings of HPDC'17 , Washington D.C.,USA, June 2017 (HPDC'17),* 2 pages.
DOI: 10.475/123_4

The Multi-Component Application Placement Problem. The MEC paradigm was introduced to solve the inefficiencies of mobile cloud computing (i.e., high latencies due to the communication between mobile devices and data-centers) by enabling data processing at the edge of the network. One of the main challenges in MEC is determining an efficient placement of the components of a mobile application on the edge servers that minimizes the cost incurred when running the application. This is the *multi-component application placement problem* (MCAPP-IM). To formulate this problem we consider a time slotted system, where the locations of users may change from one time slot to another. The location of a user is specified by its coordinates in a two-dimensional grid of cells. A user can change its location between two time slots, that is, it can move into any of the neighboring cells or stay in the same cell. The edge system is composed of a set of servers, $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$. Servers can be edge or core cloud servers having different computational powers, and therefore, different costs. Servers can be located in any cell of the two-dimensional grid and their positions are fixed. Each application C of a user consists of a set of components, $C = \{C_1, C_2, \ldots, C_n\}$. We do not impose any restrictions on the communication between the components, any component can communicate with any other component of the application (i.e., the graph modeling the application is not restricted). We also assume that a server can communicate with any other server, incurring different costs for different servers. The *objective* is to find a mapping between components and servers, such that the total placement cost is minimized. The total placement cost in time slot t is composed of four types of costs: (i) the cost of running component C_j on server S_i; (ii) the cost of relocating component C_j from server S_i to server $S_{i'}$; (iii) the communication cost between component C_j (assigned to server S_i) and the user; and, (iv) the communication cost between components C_j and $C_{j'}$ that are located on servers S_i and $S_{i'}$, respectively. We formulated the MCAPP problem as a Mixed Integer Linear Program (MILP) and use the optimal solution obtained by solving it as a lower bound on the performance of the proposed online algorithm. Due to space limitations we will not present the MILP here.

MCAPP-IM Online Algorithm. Our formulation of MCAPP problem departs from the existing work since it does not impose any restrictions on the topology of the graphs characterizing both the applications and the physical resources. Considering this general setting, we design a heuristic algorithm that solves the online version of MCAPP. In the online version of MCAPP, the values of the cost parameters introduced in the previous section may change every time slot and are not known *a priori*. We assume that at the end of every time slot the values of the parameters for the next slot are known and that the proposed algorithm determines the allocation for the next time slot based on those new values. The proposed on-line algorithm MCAPP-IM (where IM stands for Iterative Matching) is executed every time slot and consists of two phases. In *the first phase*, it determines the best matching between components of the application and the edge/core servers without considering the communication requirements among the components. In *the second phase*, it takes into account the communication costs between the components and performs a local search procedure that determines the final solution to MCAPP. MCAPP-IM algorithm has low time complexity

HPDC'17, Washington D.C.,USA
2017. 123-4567-24-567/08/06…$15.00
DOI: 10.475/123_4

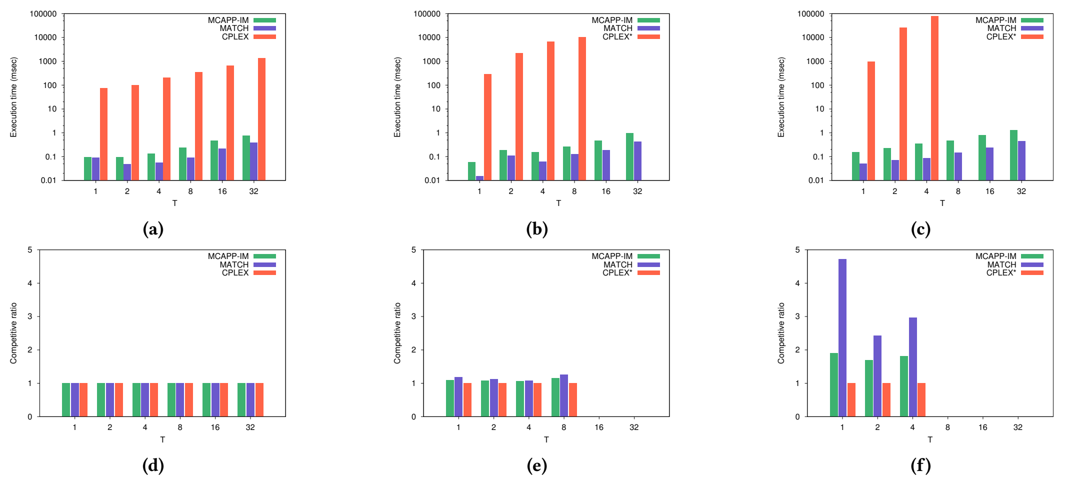


Figure 1: Execution time and competitive ratio vs. number of time slots: Computation-intensive case ((a) and (d)); Balanced communication-computation case ((b) and (e)); Communication-intensive case ((c) and (f)). (*CPLEX was not able to determine the solution for $T = 16$ and 32 in (b) and (e), and for $T = 8$, 16 and 32 in (c) and (f) in feasible time, and thus, there are no bars in the plots for those cases)

(cubic in both the number of servers and the number of components) and adds a negligible overhead to the placement and execution of the applications.

Experimental Results. We compare the performance of our algorithm, MCAPP-IM, with that of an algorithm called MATCH and with that of the optimal solution obtained by solving the MILP formulation of MCAPP. The MATCH algorithm performs matching without taking the communication among instances into account. We compare with this algorithm in order to investigate the improvement in the quality of the solution due to considering the communication time among applications in the local search phase of MCAPP-IM.

We consider that the users and servers are located within a two-dimensional grid of 150×150 cells. Initially, a user can be in any cell of the grid network and its location is drawn randomly from a uniform distribution over the locations of the grid. We assume that the mobility of users is based on the random walk model in a two-dimensional space, which is an approximation of real world mobility traces. The servers are located within the same two-dimensional grid network and the coordinates of their positions are drawn from a uniform distribution. We generate several problem instances and for each type of instance, we execute MCAPP-IM and MATCH algorithms ten times. The performance of MCAPP-IM is evaluated by computing the *actual competitive ratio* which is defined as the ratio between the value of the solution obtained by an online algorithm and that of the optimal solution for the offline problem. The optimal solution is obtained by solving the MILP formulation of MCAPP with the CPLEX-12 solver. The MCAPP-IM and MATCH algorithms are implemented in C++ and the experiments are conducted on an Intel 1.6GHz Core i5 with 8 GB RAM system.

We investigate the performance of the MCAPP-IM algorithm in terms of actual competitive ratio and execution time on a set of medium size instances consisting of $m = 10$ servers and $n = 4$ components. We chose this type of instances in order to be able to solve them optimally using CPLEX and compare the performance of our algorithm with that of the optimal solution. We consider three types of instances, computation-intensive, computation-communication balanced, and communication-intensive. In Figures 1 (a)-(c), we plot the total execution times (i.e., the sum of the execution times of all time slots) obtained by MCAPP-IM, MATCH, and CPLEX on those instances for different values of the total number of time slots, T, using a logarithmic scale. The execution time of CPLEX is several orders of magnitude higher than the execution times of both MCAPP-IM and MATCH for all three types of instances. The execution times of our proposed algorithm, MCAPP-IM, are under 1 millisecond in all cases, making it very suitable for deployment in real MEC systems. The MATCH algorithm obtains a slightly lower execution time than MCAPP-IM but as we will show next, this small execution time is obtained at the expense of not being able to provide near optimal solutions to the problem. The reason behind this is that MATCH does not take into account the communication between components when determining the placement, requiring less time than MCAPP-IM. In Figure 1 (d)-(e), we plot the actual competitive ratios obtained by the algorithms. Since CPLEX obtains the optimal solution, we plot its competitive ratio as 1 in those plots. In the case of computation-intensive instances, the actual competitive ratios obtained by MCAPP-IM and MATCH are very close to 1, thus both algorithms obtain optimal solutions or solutions that are very close to the optimal. If there is almost no communication among the components, MCAPP-IM behaves similarly to MATCH, that is the local search step is not actually able to improve the solution beyond that obtained by matching. In the case of communication-intensive instances, MCAPP-IM obtains much better competitive ratios than MATCH for all instances. That means that MCAPP-IM is able to obtain solutions that are closer to the optimal solution than those obtained by MATCH. Another important observation is that the actual competitive ratios obtained by MCAPP-IM are less than 2 for all the communication-intensive instances considered here, that is they are independent on the number of slots we considered.

The experimental results show that MCAPP-IM obtains solutions that are very close to the optimal and requires very low execution time. For the average size instances, the ones we expect to encounter in practice, the proposed algorithm performs very well with respect to both the quality of the solutions and the execution time.

Efficient Placement of Multi-Component Applications in Edge Computing Systems

{ Tayebeh Bahreini and Daniel Grosu } Dept. of Computer Science, Wayne State University

Motivation

- **Mobile Computing**
  - Many mobile applications need to perform heavy computations.
  - Some mobile applications consume a large amount of energy.

  Challenges:
  - ∗ Limited computation/storage resources
  - ∗ Limited battery life
- **Mobile Cloud Computing**
  - Computation and storage are moved from mobile devices to resource-rich servers located in clouds.

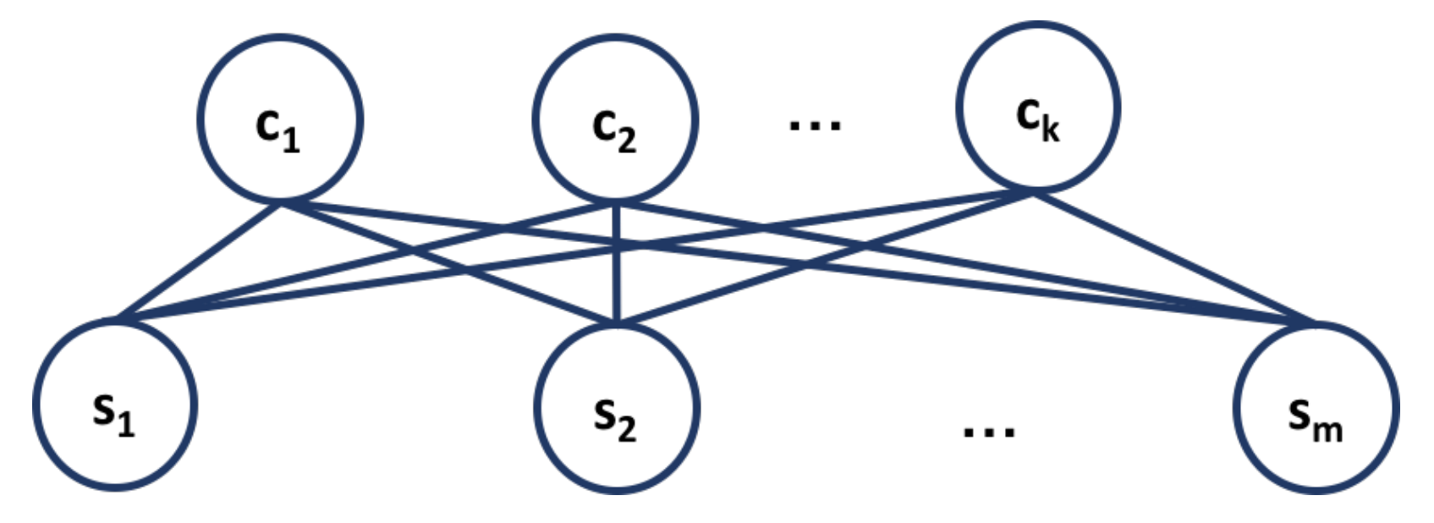
  Challenges:
  - ∗ High communication latency (limitation on wireless internet bandwidth)
  - ∗ Inefficient for applications that need a quick response time or have a large amount of data transmission
- **Mobile Edge Computing (MEC)**
  - Reduces the response time of mobile applications by allowing them to perform their computation at the edge of the network.
  - The edge can be any computing resource of the network.
- **Contributions**
  - This research addresses the *multi-component application placement problem* (MCAPP) in MEC in order to minimize the total cost of running services.
  - In the formulation of the problem, the dynamism of users' location, the network's statistics, and different cost elements are considered.
  - An efficient on-line algorithm for this problem is developed and its performance is analyzed.
  - Experimental results show that the algorithm requires very small execution times and obtains near optimal solutions

MCAPP

- **Problem Setting**
  - A time slotted system in which the location of users and the network's statistics may change from one time slot to another.
  - A set of servers $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$ that are distributed uniformly in a mesh network.
  - A mobile application with a set of components $\mathcal{C} = \{C_1, C_2, \dots, C_n\}$.
  - Any component can communicate with any other components (the graph modeling the application is not restricted).
  - A server can communicate with any other servers, incurring different costs for different servers.
- **Objective**
  - Find a mapping between components and servers, such that the total placement cost is minimized.
  - Placement costs come from different sources:
    - ∗ the cost of running component C_j on server S_i.
    - ∗ the cost of relocating component C_j from server S_i to server $S_{i'}$.
    - ∗ the communication cost between component C_j (assigned to server S_i) and the user.
    - ∗ the communication cost between components C_j and $C_{j'}$ that are located on servers S_i and $S_{i'}$, respectively.

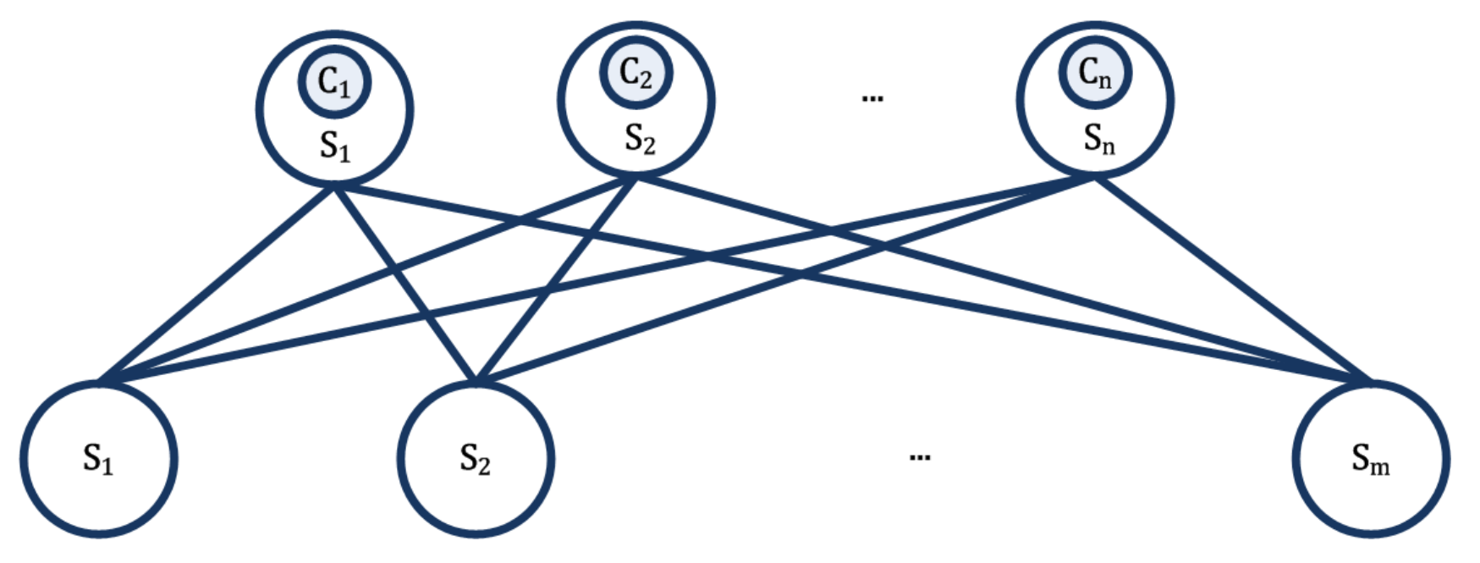
MCAPP-IM Online Algorithm

- **MCAPP and Matching**
  - When there is no inter-component communication, the placement problem in each time slot is equivalent to the matching problem.
  - In the first time slot, the problem is to match components to servers.



Matching components to servers in the first time slot

  - In other iterations, the decision maker decides whether a component stays on the current server or migrates to another one.



Matching components to servers (in time slots t > 0)

- **MCAPP-IM Algorithm**

  {Executed every time slot}

  (i) Solve the matching problem using the Hungarian algorithm without considering the inter-component communication.

  (ii) Call the L-SEARCH algorithm which takes the inter-component communication cost into account to find the final solution to MCAPP.

- **L-SEARCH Algorithm**

  Do

  (i) Find the bottleneck component that has the maximum cost of communication with other components.

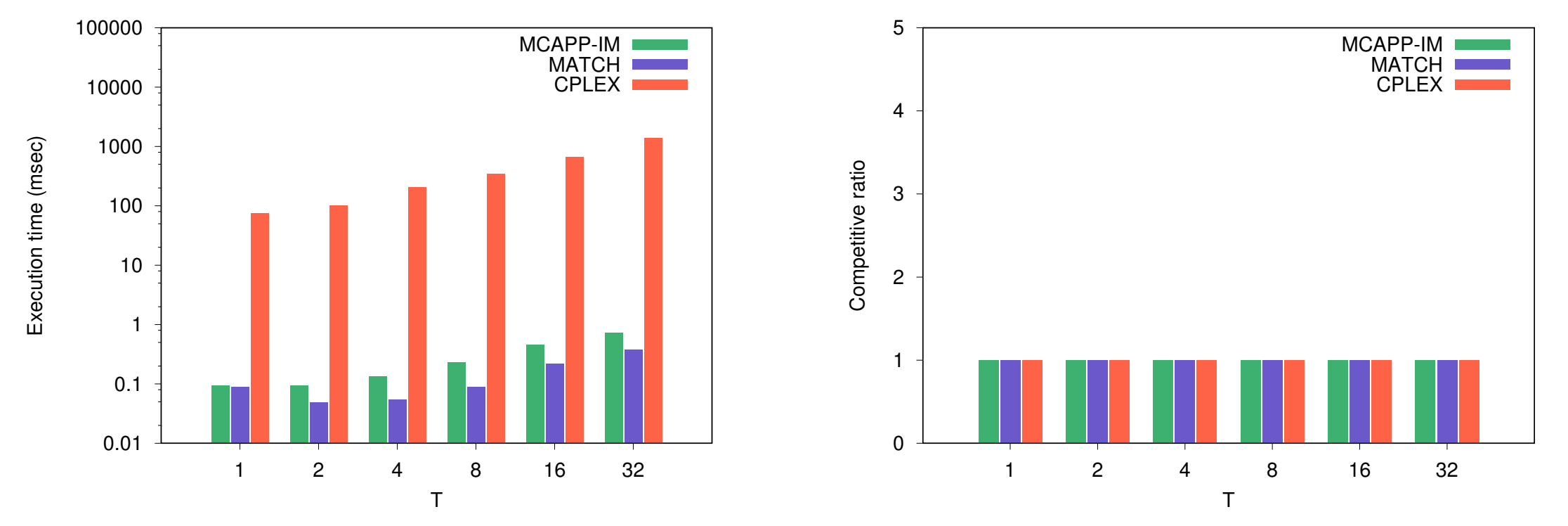
  (ii) Swap the bottleneck component with components that are placed on other servers in order to to find a lower total assignment cost for the system.

  While there is improvement

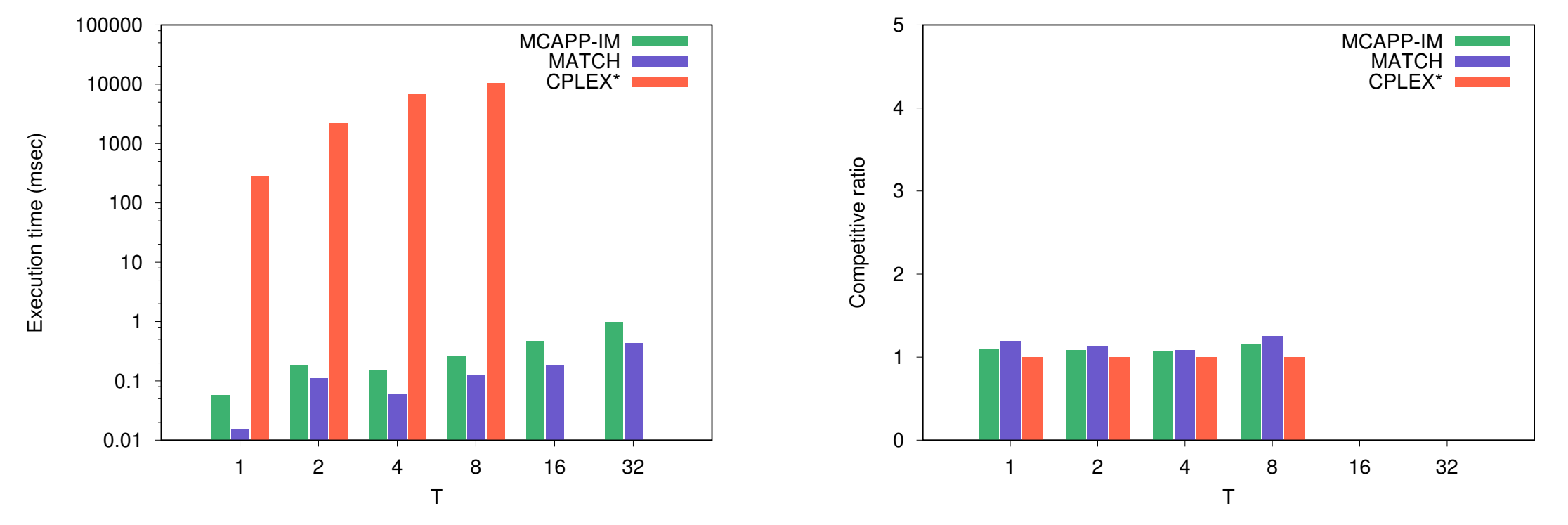
Experimental Results

- **Findings**
  - The performance of MCAPP-IM is compared with MATCH and the optimal solution obtained by CPLEX.
  - The MATCH algorithm performs matching without taking the communication among components into account.
  - MCAPP-IM obtains solutions that are very close to the optimal solution and requires a reasonable execution time.
  - The quality of solutions is relatively insensitive to the value of T (The maximum time required to run an application).

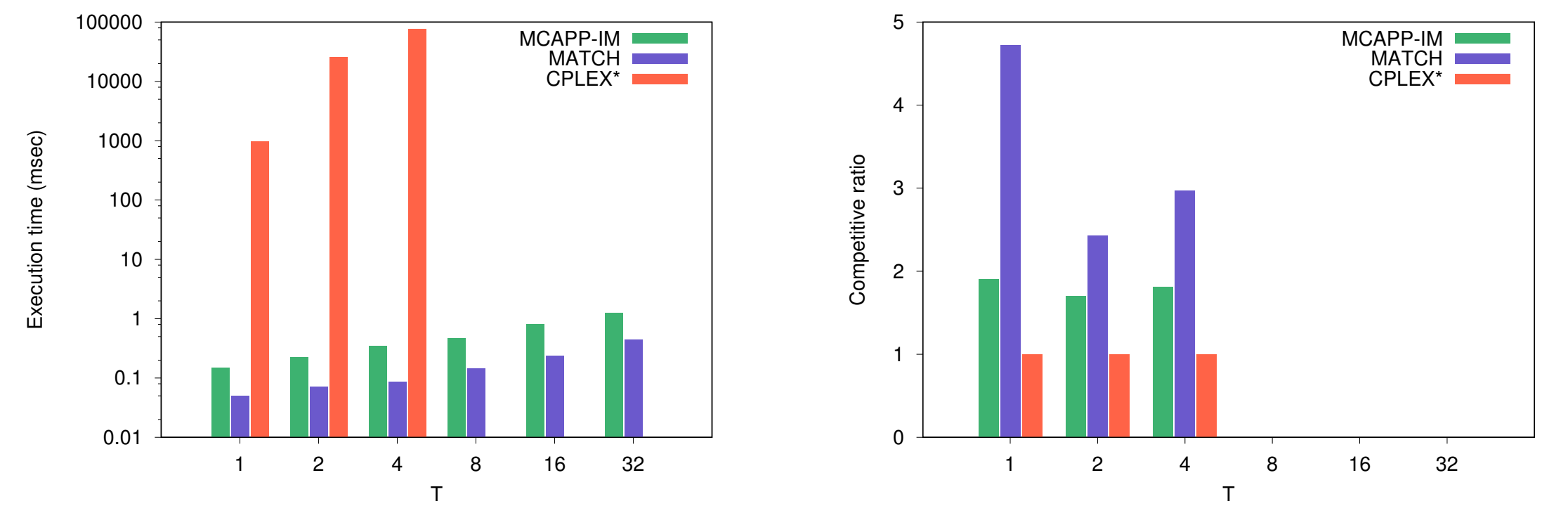
Execution time and competitive ratio vs. T



(a) Computation-intensive case



(b) Balanced communication-computation case



(c) Communication-intensive case

$$\text{Competitive ratio} = \frac{\text{solution obtained by MCAPP-IM}}{\text{solution obtained by CPLEX}}$$

Contact Information

Tayebeh Bahreini (tayebeh.bahreini@wayne.edu)
Ph.D. student - Department of Computer Science, Wayne State University