Quality-Aware Service Selection for Service-based Systems Based on Iterative Multi-Attribute Combinatorial Auction

Qiang He, Jun Yan, Hai Jin, Yun Yang

Abstract—The service-oriented paradigm offers support for engineering service-based systems (SBSs) based on service composition where existing services are composed to create new services. The selection of services with the aim to fulfil the quality constraints becomes critical and challenging to the success of SBSs, especially when the quality constraints are stringent. However, none of the existing approaches for quality-aware service composition has sufficiently considered the following two critical issues to increase the success rate of finding a solution: 1) the complementarities between services; and 2) the competition among service providers. This paper proposes a novel approach called Combinatorial Auction for Service Selection (CASS) to support effective and efficient service selection for SBSs based on combinatorial auction. In CASS, service providers can bid for combinations of services and apply discounts or premiums to their offers for the multi-dimensional quality of the services. Based on received bids, CASS attempts to find a solution that achieves the SBS owner's optimisation goal while fulfilling all quality constraints for the SBS. When a solution cannot be found based on current bids, the auction iterates so that service providers can improve their bids to increase their chances of winning. This paper systematically describes the auction process and the supporting mechanisms. Experimental results show that by exploiting the complementarities between services and the competition among service providers, CASS significantly outperforms existing quality-aware service selection approaches in finding optimal solutions and guaranteeing system optimality. Meanwhile, the duration and coordination overhead of CASS are kept at satisfactory levels in scenarios on different scales.

Index Terms—Service-based system, combinatorial auction, quality of service, service composition, service selection, Web services, integer programming.

1 INTRODUCTION

COMPUTING is evolving from component-based to service-based with a rapid progress in services research and practice. The service-oriented paradigm is emerging as a new way to engineer software systems that are composed of and exposed as services for use through standardised protocols, such as WSDL [16], UDDI [17] and SOAP [27]. Service-based systems (SBSs) are pushing traditional software engineering problems - such as requirements, specification, distribution, componentisation, composition, verification, and evolution - to their extremes. A great advantage of the service-oriented paradigm is its support for service composition. Through service composition, SBS designers can compose existing services in the form of business processes to construct new SBSs [4, 60]. BPEL [30, 43] is a de facto standard to specify the fashion in which Web services interact in a business process. Semantics [32, 33, 48] and artificial intelligence (AI) techniques [28, 45] are also popular in the area of service composition.

Service composition of an SBS usually goes through four phases: 1) planning: the SBS designer determines what functionalities are needed to realise the SBS; 2) service discovery: the SBS designer looks up candidate services that can provide the required functionalities; 3) service selection: the SBS designer selects a subset of the candidate services for the SBS. The selected services must achieve the SBS owner's optimisation goal for the SBS while fulfilling all constraints for the quality of the SBS, e.g., response time, availability, etc.; 4) service delivery: the selected services are executed in a certain order to realise the SBS.

With the development and popularity of e-business, e-commerce, especially the pay-as-you-go business model promoted by Cloud computing [26], there are more and more functionally-equivalent services available at different quality levels. According to ProgrammableWeb, an online Web service directory, the number of published Web services has quadrupled since 2009. The statistics published by webservices.seekda.com, a Web service

- *Qiang He is with the Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China and the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia 3122. E-mail: heqiang@gmail.com.*
- *Jun Yan is with the School of Information Systems and Technology, University of Wollongong, Wollongong, Australia 2522. E-mail: jyan@uow.edu.au.*
- *Hai Jin is with the Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: hjin@hust.edu.cn.*
- *Yun Yang is with the School of Computer Science and Technology, Anhui University, Hefei, Anhui 230039, China and the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia 3122. E-mail: yyang@swin.edu.au.*

Manuscript received (insert date of submission if desired). Please note that all acknowledgments should be placed at the end of the paper, before the bibliography.

search engine, also indicate an exponential growth in the number of published Web services in the past few years. Driven by the widespread of Cloud computing, the service-oriented environment is moving rapidly toward a perfect competition environment in which service providers compete for service contracts [24], i.e., the contracts for provision of services. In such an environment, service selection for SBSs is a complex decision making process which involves a number of stakeholders. On one hand, as buyers in the market, SBS designers can benefit from exploiting the competition among service providers. The competition among service providers increases the success rate of finding a solution in severe scenarios where severe quality constraints are imposed on the SBS. The fiercer the competition is for a service contract, the more likely the SBS designer would be able to obtain satisfactory offers for the price and quality of the service. Thus, the competition can increase the system optimality, i.e., the degree to which the SBS owner's optimisation goal is achieved. Therefore, in the service selection phase, the SBS designer should negotiate service level agreements (SLAs) with multiple candidate service providers and then select the best offer for each service [22, 39, 56]. On the other hand, as sellers in the market, service providers' profits often increase when the numbers of service contracts that they win increase. This is a strong incentive for service providers that can provide multiple types of services to propose competitive quality-of-service (QoS) offers, i.e., offers that specify promised QoS, in order to win more service contracts. Furthermore, services often show complementarity [20] – they can be provided at better QoS levels together by a single provider than by multiple individual providers. Think of two different scenarios of a double-layer encryption SBS composed of two services that use different encryption schemes. When the two services are provided by a single provider rather than two individual providers, shorter response time is expected from the SBS because both encryption operations are performed by the service provider in-house without having to transmit the intermediate data across organisational boundaries. In addition, less cost of service usage for the SBS, i.e., the total cost of using the services that compose the SBS, is expectable because a discount for the two services as a bundle offered by the service provider is potentially available. Thus, service providers capable of providing complementary services will be able to gain advantages over other service providers during the SLA negotiation as they can offer better QoS at lower prices.

Due to the above characteristics, service selection for an SBS based on SLA negotiation can be very complicated – it is in fact a NP-complete problem [8] – because the selected services must achieve the SBS owner's optimisation goal while fulfilling all quality constraints for the SBS. Thus, we need an effective and efficient approach that addresses the following key issues:

1. Some service providers have the ability to provide multiple types of services. They may wish to compete for multiple service contracts in the SLA negotiation and their QoS offers (i.e., offers for the quality of the services) are dependent on how many of them that they won. Those service providers should be able to flexibly express their QoS offers for combinations of services.
2. When a solution cannot be found, e.g., the quality constraints for the SBS cannot be fulfilled by service providers' current QoS offers, an additional round of SLA negotiation is needed for service providers to improve their offers. Guidance on how to improve their offers must be provided to: 1) exploit the competition between the service providers; and 2) coordinate the SLA negotiation.
3. The NP-complete service selection problem for SBS is subject to computational uncertainty in large-scale scenarios. The SLA negotiation must remain effective and efficient when the service composition scenario scales up.

It has been long proven that auction, a form of one-to-many negotiation, is effective and efficient in a perfect competition environment [40, 55], especially when the items to be allocated exhibit complementarities [20]. In many e-commerce cases, such as Amazon.com, eBay.com and Overstock.com, auction has been proven to be able to well capture the preferences of both buyers and sellers, and to ensure their satisfaction. However, traditional single-item auctions are unsuitable for service composition scenarios where multiple abstract services are involved. This paper presents a novel approach called Combinatorial Auction for Service Selection (CASS) to support effective and efficient quality-aware service selection based on combinatorial auction – a type of auction where bidders can bid for multiple items [20]. In CASS, the abstract services of the SBS are auctioned as *items*. Service providers, as *bidders*[1] in the auction, can place bids (i.e., QoS offers) for the services. If a solution is found based on the bids, the SBS designer, as *auctioneer*[2] in the auction, awards the service contracts to the winning bidders who are responsible for delivering concrete services in the service delivery phase according to negotiated SLAs. CASS can facilitate effective and efficient SLA negotiation by exploiting service providers' QoS capacities, i.e., capacities for QoS provision, and eliciting competition among them. The key features of CASS are as follows:

1. CASS allows service providers to bid for combinations of services in a structured manner. This feature allows service providers to utilise their QoS capacities for complementary services. They can apply discounts or premiums to their QoS offers when they are bidding for combinations of services.
2. CASS allows SBS designers to negotiate with multiple candidate service providers for each constituent service of the SBSs. This creates competition among service providers.
3. When a solution cannot be found, e.g., the current QoS offers proposed by service providers cannot fulfil the quality constraints for the SBS, the auc-

[1] In the context of this research, terms "bidder" and "service provider" are interchangeable.

[2] In the context of this research, terms "auctioneer" and "SBS designer" are interchangeable.

tion iterates, allowing the service providers to improve their bids until a solution is found or their QoS capacities are exhausted. A novel approach for the generation of multi-dimensional *ask-QoS* - the offer for QoS asked by the auctioneer – is designed to help SBS designers coordinate the auction processes and exploit the competition among service providers. Ask-QoSs provide guidance to the bidders on analysing their positions in the competition and improving their bids.

4. CASS models the winner determination problem (WDP) in combinatorial auction as a Constraint Optimisation Problem (COP) and adopts Integer Programming (IP) techniques to solve the WDP. In the COP model, SBS designers can specify various optimisation goals and different constraints flexibly, including the quality constraints for the SBSs and the auction constraints.

Based on the current state of the practice for QoS-aware service selection, including negotiation and IP techniques, CASS takes a further step by exploiting the complementarities between services and the competition between service providers. As the service-oriented environment becomes more complex and competitive, CASS offers a promising solution to finding services with satisfactory QoS for constructing SBSs in the service-oriented environment.

The rest of the paper is organised as follows. The next section analyses the requirements with a motivating example. Section 3 introduces some preliminaries. Section 4 introduces the compositional quality model adopted in this research. Section 5 presents the procedure of CASS, followed by supporting mechanisms described in Section 6. Section 7 presents experimental evaluation and validity analysis. Section 8 reviews related work. Finally, Section 9 summarises the major contributions of this paper and outlines future work.

2 Requirements Analysis with Motivating Example

This section presents an example SBS to illustrate the motivation and requirements of this research. This example originates from [3] and is adapted to the characteristics of this research. As shown in Fig. 1, the business process of this SBS includes six abstract services: *New Car Search*, *Used Car Search*, *Best Car Selection*, *Insurance Quote*, *Loan Quote* and *Result Merging*. In order to realise the SBS, six concrete services are needed to implement the six abstract services.

The SBS serves for its clients by processing their requests. The clients submit requests to the SBS, specifying their criteria for selecting cars, e.g., brand, type and model. In response to each request, the SBS returns a list of new or used cars with a loan offer and an insurance quote for each car on the list.

The service selection for this SBS must address the following issues properly.

The provider of the SBS often has quality constraints for the SBS. For example, the SBS must be able to process

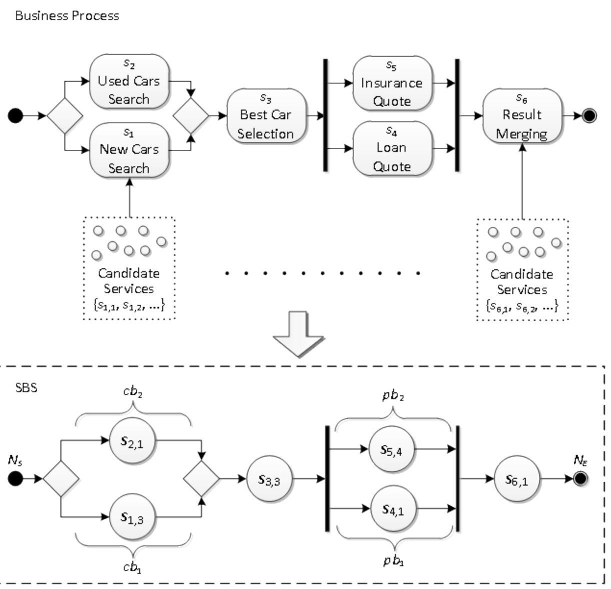


Fig.1. An example SBS.

5000 requests per second – the throughput at peak times – and the response time of the SBS must not exceed 4 seconds. To fulfil those quality constraints for the SBS, the SBS designer needs to negotiate with candidate service providers, and from the available QoS offers, select the "right" one for each abstract service. Similar to other research efforts [2-4, 58, 60, 61], in this research, we assume that alternative functionally equivalent services are available and can be categorised into different *service classes* based on their functionalities.

Besides the quality constraints for the SBS, the SBS owner usually has an optimisation goal for the SBS. Different SBS owners may have different optimisation goals, depending on their business needs. Two examples of such goals are 1) to maximise the overall system performance regardless of the cost of service usage; and 2) to minimise the cost of service usage for the SBS. The selected concrete services must achieve the SBS owner's optimisation goal for the SBS. In Fig. 1, suppose the SBS owner requires minimal cost of implementing the SBS, the SBS designer needs to make sure that the selected service providers offer the cheapest services while collectively meeting all the quality constraints for the SBS.

The available QoS offers for an abstract service often differ in their QoS values. For example, a *Used Car Search* service provided by a service provider, e.g., $s_{1,1}$, returns the search results faster than another *Used Car Search* service provided by a different service provider, e.g., $s_{1,3}$, but requires a higher price.

Some service providers may be capable of providing multiple services, e.g., the *New Cars Search*, *User Cars Search* and *Best Car Selection* services. More importantly, they can offer better quality offers for these services because these services often exhibit complementarity – they can be provided at better QoS levels by a single provider than multiple individual providers. For example, the overall response time of the *Best Cars Search*, *User Cars Search* and *Best Car Selection* services can be reduced if they are provided by the same provider because the results from executing the car search services can be immediately processed by the *Best Car Selection* service without cross-organisational data transmission. Since such "multi-functional" providers are capable of providing more services (and usually willing to for more profit), discounts on the prices or premiums for the quality of the services are negotiable between the SBS designer and the service provider. For example, a service provider may charge a lower price for the *Insurance Quote* service if the SBS designer also uses its *Loan Quote* service. Thus, the service providers should be able to express such discounts and premiums in their QoS offers flexibly. Those potentially available discounts and premiums increase the possibility of finding appropriate services for the SBS, especially when the quality constraints for the SBS are severe.

Sometimes, a solution for the SBS cannot be found based on the current QoS offers. For example, using even the cheapest services still exceeds the SBS owner's budget for the SBS. In such cases, the service providers should be allowed to improve their offers. Then, based on the improved offers, the SBS designer can try to find a solution again. In addition, to make sure that the negotiation proceeds in a desirable direction towards a satisfactory solution, guidance should be provided to the service providers on how to improve their bids in order to be included as part of the final solution.

3 Preliminaries

In this research, we adopt combinatorial auction to manage the complex negotiation between SBS designers and service providers over multi-dimensional quality of multiple services. In the auction, service providers use a specific bidding language to propose QoS offers as bids for the SBS designer's selection. The winning bids will determine which concrete services are selected for constructing the SBS. In this section, the concepts of combinatorial auction and bidding language are presented.

3.1 Combinatorial Auction

Combinatorial auctions are auctions where bidders can bid for combinations of items [20]. It has been proven that combinatorial auctions can lead to very effective allocation of multiple items [50]. There are already numerous successful examples of combinatorial auctions in practice, e.g. railroad tracks [10], real estate [47], pollution rights [34], airport time slot [49], distributed scheduling of machine time [54] and advertising space [23]

In a combinatorial auction, a seller sells m items to n bidders. Let $K=\{s_1, s_2, \ldots, s_m\}$ be the set of items and $\mathcal{B}$ be the bids submitted by the bidders. The notation and acronyms adopted in the paper are summarised in the Appendix. A bid is represented by a tuple (S_i, p_i), $S_i \neq \emptyset$ and $S_i \subseteq K$, where S_i represents the set of items that the bid is placed on and p_i is the price for S_i. The key problem of the combinatorial auction is winner determination, i.e., to select a subset $\{(S_1, p_1), (S_2, p_2), \ldots, (S_n, p_n)\}$ of $\mathcal{B}$ that fulfils the following two constraints:

$$S_1 \cup S_2 \cup \ldots \cup S_n = K \qquad (1)$$

$$\forall S_i, S_j,\ i \neq j \Rightarrow S_i \cap S_j = \emptyset \qquad (2)$$

and meanwhile, maximises the auction revenue:

$$\text{maximise} \sum_{i=1}^{n} \mathit{profit}(S_i) \qquad (3)$$

where p_i is the profit obtained from S_i.

Constraint (1) ensures that all items are covered by the selected bids. Constraint (2) ensures that every item is included in only one selected bid.

The advantage of combinatorial auctions is that the bidders can fully express their preferences for different items. This is particularly important when the items are complementary.

In a service composition scenario, an SBS designer needs to select a set of services from the candidate services for an SBS S that achieves the SBS owner's optimisation goal, while fulfilling all quality constraints for the SBS. The SBS designer can hold a reverse multi-attribute combinatorial auction where service providers propose bids to compete for the service contracts, i.e., the contracts for the provision of the services of S. At the end, the service contracts are awarded to the final winning bidders. In such an auction, each bid is represented by a tuple (S, Q), $S \neq \emptyset$ and $S \subseteq \mathsf{S}$, where S represents the set of abstract services on which the bid is placed, e.g., $\{s_1, s_3, \ldots\}$, and Q represents the service provider's QoS offer for S, e.g., $\{Q(s_1), Q(s_3), \ldots\}$. Besides constraints (1) and (2), the solution to the WDP needs to fulfil additional constraints – the concrete services selected based on the winning bids must fulfil all quality constraints for the SBS. Formally, given a set of quality constraints $C=\{c_1, \ldots, c_t\}$, the following constraint needs to be fulfilled:

$$\forall p \in [1,t] \Rightarrow \begin{cases} q_p(\mathsf{S}) \geq c_p & \text{for positive quality parameter} \\ q_p(\mathsf{S}) \leq c_p & \text{for negative quality parameter} \end{cases} \qquad (4)$$

where $q_p(\mathsf{S})$ is the pth quality parameter of S. The definitions of positive and negative quality parameters are given in Section 4.4.

Furthermore, the winning bids need to achieve the SBS owner's optimisation goal by maximising the auction revenue, e.g., maximising the overall utility of the SBS:

$$\text{maximise}(u(\mathsf{S})) \qquad (5)$$

where $u(\mathsf{S})$ is the function for calculating the utility of S.

3.2 Bidding Language

In a combinatorial auction, a bidder can propose multiple bids using the bidding language which specifies the structure of bids. A bidding language should allow bidders to bid in a flexible and straightforward manner. However, it should not be overly expressive because that would unnecessarily increase the complexity of the auction. For example, given a set S of n items, the total number of possible combinations of k items is $C(n, k)$, i.e., the k-combinations of S. If a bidding language requires a bidder to attach a value to each possible combination of items, a bidder has to submit a bid of size $C(n, 1)+C(n, 2)+\ldots+C(n, n)=2^n-1$, which is only practically feasible for very small m. In this research, the exclusive-or (XOR) bidding language is adopted because it is flexible and straightforward in expressing the complementarities between items and has emerged as a popular choice in recent combinatorial auction designs [9, 20]. Using the XOR bidding language, a bidder can submit bids in the form of (S_1, Q_1) *xor* (S_2, Q_2) *xor…xor* (S_n, Q_n) stating the semantics "I will buy at most one of these combinations." In the context of this research, the bidders sell services and compete for service contracts. Thus, the semantics is "I will offer at most one of these combinations of services."

4 Compositional Quality Model

The quality evaluation of an SBS is the basis for quality-aware service selection. In this section, we first present the compositional structures adopted in this research for representing the business processes and service compositions of SBSs. Based on the adopted compositional structures, we present how the SBS designer can calculate the execution probabilities of different execution scenarios of a service composition. Then, we introduce the utility and quality evaluation methods for SBSs.

4.1 Compositional Structures

Compositional structures describe the order in which the (abstract and concrete) services are performed in the business process and the service composition of SBSs. Similar to other work [4, 53, 61], in this research, we consider four types of basic compositional structures, i.e., *sequence*, *conditional branch*, *loop* and *parallel*, which are included in BPMN [44] and addressed by BPEL [43] – the de facto standards for specifying service-oriented business processes.

- **Sequence**. In a sequence structure, the services are executed one by one.
- **Conditional Branch**. In a conditional branch structure, only one branch is selected for execution. For every set of branches {cb_1, …, cb_n}, the execution probability distribution {$prob(cb_1)$, …, $prob(cb_n)$} ($0 \le prop(cb_i) \le 1$, $\sum_{i=1}^{n} prob(cb_i) = 1$) is specified, where $p(cb_i)$ is the probability that branch i is selected for execution.
- **Loop.** In a loop structure, the loop is executed for n ($n \ge 0$) times. For every loop, the probability distribution {$prob_0$, …, $prob_{MNI}$}, ($0 \le prob_i \le 1$, $\sum_{i=0}^{MNI} prob_i$ =1) is specified, where $prob_i$ is the probability that the loop iterates for i times and *MNI* is the expected maximum number of iterations for the loop.
- **Parallel.** In a parallel structure, all the branches are executed at the same time.

$p(cb_i)$, $prob_i$ and *MNI* can be evaluated based on the SBS's past executions or can be empirically specified by the SBS designer [4, 60]. Similar to [4], we require that for every loop, the *MNI* must be determined. Otherwise, if an upper bound for the number of iterations for a loop does not exist, the quality of the SBS cannot be calculated because the loop can iterate infinitely [60]. In addition, if $p(cb_i)$ and $prob_i$ are unknown, an average value will be assigned to each of the branches in conditional branch and loop structures. For example, for a conditional branch that consists of five branches, there is $prob(cb_1)=prob(cb_2)=prob(cb_3)= prob(cb_4)=prob(cb_5)=0.2$.

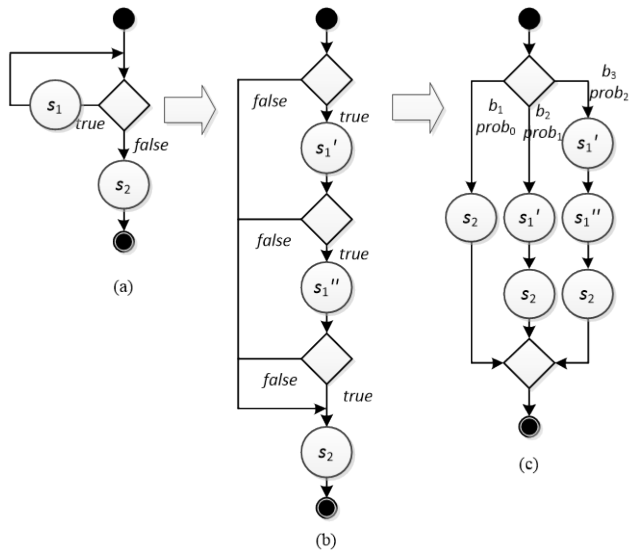


Fig. 2. The loop peeling process.

In this research, we represent the business processes and service compositions of SBSs using UML activity diagrams, where activity nodes represent services and edges represent data transmissions. We assume that the business process of an SBS is characterised by only one entry point and one exit point (e.g., N_S and N_E in Fig. 1), and it only includes *structured loops* with only one entry point and one exit point. If an SBS includes loops, we peel the loops by representing loop iterations as a set of branches with certain execution probabilities as in [25]. Fig. 2 gives an example of peeling a loop structure (*MNI*=2) by transforming it into a conditional branch structure that contains three conditional branches cb_1, cb_2 and cb_3, where p_0, p_1 and p_2 are the probabilities that cb_1, cb_2 and cb_3 are selected for execution respectively.

4.2 Execution Scenarios

In a service composition where branches or loops are involved, multiple possible *execution scenarios* of the service composition can be identified. As introduced in Section 4.1, a loop structure can be peeled and then represented by a conditional branch structure. Thus, by detecting the conditional branch structures, possible execution scenarios can be identified from the service composition as any execution scenario contains only one branch from each of the conditional branch structures. These execution scenarios do not contain branch or loop structures. As presented in Fig. 3, two possible execution scenarios can be identified from the service composition presented in Fig. 1. The quality and utility evaluation of an SBS must consider all the possible execution scenarios according to their *execution probabilities*, i.e., the appearance probabilities of the execution scenarios. Therefore, we need to calculate the execution probability of each possible execution scenario identified from the service composition. The execution probability of an execution scenario is determined by the execution probabilities of all its execution paths. Whether an execution path is selected for execution is dependent

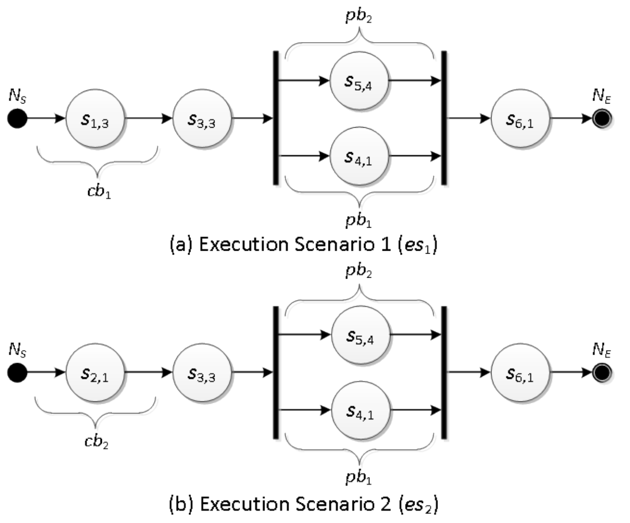


Fig. 3. Example execution scenarios.

on the decisions made at the conditional branches. Thus, the *execution probability of an execution scenario* is the product of the execution probabilities of all the conditional branches in the execution scenario. For an execution scenario es_e, the execution probability of es_e is calculated as:

$$ep(es_e)=\prod_{i=1}^{n} prob(cb_i) \tag{6}$$

where cb_i represents the conditional branches that es_e includes.

For example, in Fig. 3, there is $ep(es_1)=prob(cb_1)$ and $ep(es_2)=prob(cb_2)$.

4.3 Utility Evaluation

Functionally equivalent services usually differ in multiple QoS parameters. For example, in Fig. 1, the *Insurance Quote* service provided by a service provider may have shorter response time but require a higher price than the *Insurance Quote* service provided by another service provider. Which is better depends on the client's preference and priority. Selection from these services by their QoS characteristics is a multi-attribute decision making problem. For the purpose of ranking and sorting the services in a same service class (i.e., the group of services that provide the same functionality), a method is needed to evaluate a given service based on its multiple QoS parameters. In this research, we use a utility function for service evaluation by applying the Simple Additive Weighting (SAW) technique [57], one of the most widely used techniques to obtain an overall score from multiple dimensions [2-4, 60].

First, the raw QoS values are normalised to remove the incomparability between the units of measurement for different QoS parameters:

$$u(q_p(s_{i,j}))=\begin{cases}\dfrac{q_p^{max}(SC_i)-q_p(s_{i,j})}{q_p^{max}(SC_i)-q_p^{min}(S_i)} & \text{if } q_p^{max}(SC_i)\neq q_p^{min}(SC_i)\\ 1 & \text{if } q_p^{max}(SC_i)=q_p^{min}(SC_i)\end{cases} \tag{7}$$

where $q_p^{max}(SC_i)$ and $q_p^{min}(SC_i)$ are the maximum and minimum values, respectively, for the pth QoS parameter in the ith service class, and $q_p(s_{i,j})$ is the value of the pth QoS parameter of the jth service in the ith service class.

To accommodate non-numerical QoS parameters, such as reputation that are expressed by a rating selected from {very high, high, medium, low, very low}, the approach proposed in [42] is adopted. Based on a pre-defined semantics-based hierarchical structure of all possible values of a non-numerical QoS parameter, each level of the hierarchy is associated with a numerical value. In this way, we can calculate the utility of the QoS parameter and term the QoS parameter negative or positive. If the levels that are more preferable to end-users are assigned with higher values, the QoS parameter is treated as a positive QoS parameter, and vice versa.

After the normalization, the utility of service s with t QoS parameters can be calculated by:

$$u(s)=\sum_{p=1}^{t} w_p \times u(q_p(s)) \tag{8}$$

where w_p ($w_p\epsilon[0, 1]$ and $\sum_{p=1}^{r} w_p = 1$) is the weight that represents the SBS owner's preference and priority for the pth QoS parameter of the SBS.

Now the utility of an execution scenario es composed by n ($n\geq1$) services $s_1, \ldots, s_n$, can be calculated by:

$$u(es)=\sum_{i=1}^{n} u(s_i) \tag{9}$$

Since an SBS may contain multiple execution scenarios the utility calculation of an SBS $\mathbf{S}$ must consider all the execution scenarios $es_1, \ldots, es_n$ according to their execution probabilities.

$$u(\mathbf{S})=\sum_{e=1}^{n} ep_e \times u(es_e) \tag{10}$$

where ep_e is the execution probability of es_e. The more frequent an execution scenario is, the more it contributes to the utility of the SBS.

4.4 Probabilistic Quality Evaluation

The quality of an SBS depends on the quality of its constituent services and the paths selected for execution. For example, the response time of an SBS at runtime depends on the longest execution path in the execution – the execution path with the maximum execution time. An execution path is a set of services forming a sequential path from the initial service to the final service of a service composition. Take Fig. 1 as an example, there are four execution paths: EP_1=$s_{2,1}$-$s_{3,3}$-$s_{5,4}$-$s_{6,1}$, EP_2=$s_{2,1}$-$s_{3,3}$-$s_{4,1}$-$s_{6,1}$, EP_3=$s_{1,3}$-$s_{3,3}$-$s_{5,4}$-$s_{6,1}$ and EP_4=$s_{1,3}$-$s_{3,3}$-$s_{4,1}$-$s_{6,1}$. Each execution path is selected with a probability. Different combinations of execution paths lead to different execution scenarios, and consequently, different QoS performance. Thus, the probabilistic quality evaluation of an SBS requires aggregating the quality of its constituent services, considering all the execution scenarios of the service composition according to their execution probabilities. For an SBS $\mathbf{S}$, let es_e (e=1, 2, …) be the execution scenarios of the service composition of $\mathbf{S}$, EP_i (i=1, 2, …) be the execution paths in the service composition, $s_{i,j}$ (j=1, 2, …) be EP_i's constituent services, the overall quality of $\mathbf{S}$ can be evaluated using the aggregation functions presented in Table 1 – an extended version of the QoS aggregation functions intro-

TABLE 1
QUALITY AGGREGATION FUNCTIONS

Quality Parameter	Aggregation Function
Cost	$q_{cost}(\mathbf{S}) = \sum_{s_j \in \mathbf{S}} q_{price}(s_j)$
Response Time	$q_{rt}(es_e) = \max_{EP_i \in es_e} \left(\sum_{s_{i,j} \in EP_i} q_{rt}(s_{i,j}) \right)$ $q_{rt}(\mathbf{S}) = \sum_{es_e \in \mathbf{S}} ep(es_e) \times q_{rt}(es_e)$
Availability	$q_{ava}(es_e) = \prod_{s_j \in es_e} q_{ava}(s_j)$ $q_{ava}(\mathbf{S}) = \sum_{es_e \in \mathbf{S}} ep(es_e) \times q_{ava}(es_e)$
Throughput	$q_{tp}(es_e) = \min_{EP_i \in es_e} \left(\min_{s_{i,j} \in EP_i} (q_{tp}(s_{i,j})) \right)$ $q_{tp}(\mathbf{S}) = \sum_{es_e \in \mathbf{S}} ep(es_e) \times q_{tp}(es_e)$

duced in [60]. For example, the cost of the SBS $\mathbf{S}$ can be calculated by $q_{cost}(\mathbf{S})=q_{price}(s_{1,3})+q_{price}(s_{2,1})+q_{price}(s_{3,3})+q_{price}(s_{4,1})+q_{price}(s_{5,4})+q_{price}(s_{6,1})$ and the response time of $\mathbf{S}$ can be calculated by $q_{rt}(\mathbf{S})=ep(es_1)\times q_{rt}(es_1)+ep(es_2)\times q_{rt}(es_2)$.

Numerical QoS parameters can be divided into two categories: positive and negative QoS parameters, defined as follows:

Definition 1: Positive QoS parameter. A positive QoS parameter is a QoS parameter whose evaluation will increase as its value increases. Given the utility function $u(q(s))$ for evaluating a QoS parameter q of a service s, q is a positive QoS parameter when:

$$\forall q_1, q_2 \in D(q(s)) \quad q_1 \le q_2 \Rightarrow u(q_1) \le u(q_2)$$

Availability and throughput are two typical positive QoS parameters.

Definition 2: Negative QoS parameter. A negative QoS parameter is a QoS parameter whose evaluation will decrease as its value increases. Given the utility function $u(q(s))$ for evaluating a QoS parameter q of a service s, q is a negative QoS parameter when:

$$\forall q_1, q_2 \in D(q(s)) \quad q_1 \le q_2 \Rightarrow u(q_1) \ge u(q_2)$$

Price (or cost) and response time are two typical negative QoS parameters.

In this paper, the discussion and examples are based on price (or cost) and response time, which have also been the basis for QoS consideration in other approaches [3, 4, 60]. The QoS parameters introduced in other literatures can be generalised as added dimensions in our quality model.In the remainder of this paper, we use negative QoS parameters and omit mostly repetitive yet similar introduction for positive QoS parameters.

5 CASS Procedure

CASS is an iterative auction, which allows bidders to improve their bids round by round for auctioneer to find a satisfactory solution. This section presents the procedure of CASS.

Before the auction, the SBS designer needs to look up candidate services for the abstract services in the business process of the SBS. This can be achieved by querying public UDDI registries, search engines and service portals. An alternative is to publish CASS as a service which service providers can subscribe to. The auction cannot start until at least one candidate service is found for each abstract service.

Not all the abstract services in the business process of the SBS must be included in the auction. For example, if there is only one candidate service for an abstract service or a specific service is preferred to the SBS designer, the service will be selected immediately as long as its QoS meets the corresponding quality constraints (if any).

The five steps of CASS are depicted in Fig. 4, as explained below.

Step 1: At the start of the auction, the auctioneer (i.e., the SBS designer) distributes a Request for Proposal (RFP) to the providers of the candidate services. The RFP specifies the following preliminary information about the auction: 1) the abstract services that are to be auctioned; and 2) the required QoS information that service providers must provide to participate in the auction, including the types of QoS parameters and the measurement units for the QoS. The information contained in the RFP is the basis for the service providers' first bids.

Step 2: The service providers that are interested in participating in the auction submit their bids. If the bids are valid, the service providers are officially acknowledged as bidders. A bid is considered invalid if it does not contain complete QoS information as required.

Step 3: The auctioneer solves the WDP and determines the current winning bids – the winning bids in the current

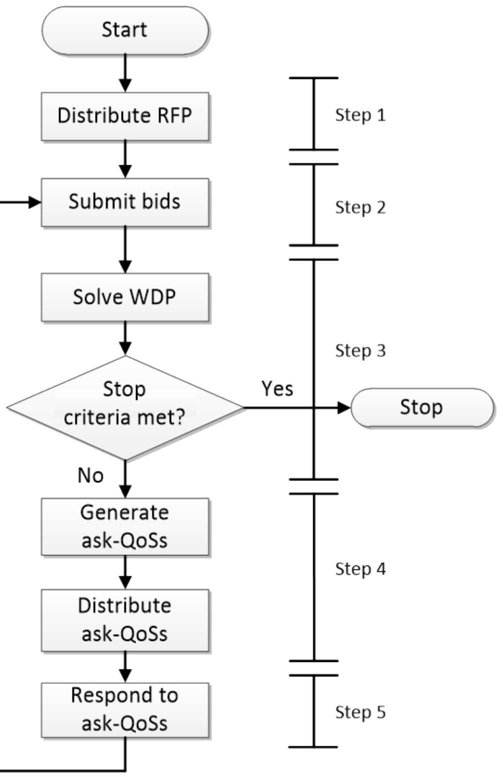


Fig. 4. The process of CASS.

round – based on received bids. The auctioneer then checks the stop criterion. If the stop criterion is met, the auction stops.

Step 4: If the criterion is not met, e.g., a solution cannot be found, the auction needs to continue. Ask-QoSs are generated and distributed to bidders, along with the current winning bids. A deadline for the bidders to propose their next bids is also sent.

Step 5: According to the received ask-QoSs and the current winning bids, the bidders can respond by 1) accepting the ask-QoS or proposing even better bids; or 2) withdraw some or all of its bids.

Steps 2 to 5 are repeated until the stop criterion is met. Sometimes a final deal-sealing round needs to be triggered to determine the final winning bidders for certain services. The details are provided in Section 6.1.3.

There are service providers that offer their services at non-negotiable QoS levels, e.g., Amazon in the current cloud market. Such service providers can still be included in the auction. Their offers will be compared with other service providers' offers. The only differences between those service providers and other service providers are twofold: 1) their offers stay unchanged in the auction until beaten; 2) they do not receive ask-QoS.

6 CASS Mechanisms

In this section, we present the mechanisms that support CASS, including bidding constraints, winner determination, stop criteria and ask-QoS generation.

6.1 Bidding Constraints

Bidding constraints are rules that bidders must follow when proposing bids in the auction. In this section, we introduce the bidding constraints imposed in CASS.

6.1.1 Non-Linear Bids

The QoS offers proposed by bidders can be non-linear. The definition of non-linear bids is as follows:

Definition 3: (Non-Linear Bids). Given bids $(\{s_m\}, \{Q_i(s_m)\})$, $(\{s_n\}, \{Q_j(s_n)\})$, $(\{s_m, s_n\}, \{Q_k(s_m), Q_k(s_n)\})$ that involve two abstract services s_m and s_n $(s_m \neq s_n)$, non-linear bids allow:

$$q_{i,p}(s_m)+q_{j,p}(s_n)\neq q_{k,p}(s_m)+q_{k,p}(s_n) \quad (16)$$

where $q_{i,p}(s_m)$, $q_{j,p}(s_n)$, $q_{k,p}(s_m)$ and $q_{k,p}(s_n)$ represent the p^{th} QoS parameter of s_m and s_n specified in $Q_i(s_m)$, $Q_j(s_n)$, $Q_k(s_m)$ and $Q_k(s_n)$.

For example, a bidder can propose \$300.00 for service s_m, \$400.00 for service s_n and \$650.00 for the combination of s_m and s_n. This allows bidders to apply discounts to their offers for negative QoS parameters (and premiums to positive QoS parameters) according to the complementarities between the services that they are bidding for.

6.1.2 Minimum Decrement

If a solution cannot be found, the auction iterates, allowing the bidders to improve their bids. In order to guarantee that the auction proceeds in the right direction towards a satisfactory solution, the auctioneer generates and sends ask-QoSs to the bidders as guidance on their bidding. Ask-QoS is an extension of ask-price which in an auction is usually defined as the price a seller of a goods is willing to accept for that particular goods. The ask-QoS in CASS specifies the QoS offer asked by the auctioneer. Corresponding to a bid, each ask-QoS is specified by (S, AQ), where S represents the set of abstract services that the bid is placed on and AQ represents the base QoS offer asked by the auctioneer from the bidder for S. Based on received ask-QoSs, bidders improve their bids to increase their chances of winning, i.e., being part of the final solution. The QoS offers specified in the bids for the next round must not be worse than the ask-QoSs. Given one of the bids proposed by a bidder br_k in the current round is $(\{s_i\}, \{\$110.00\})$. Suppose that the ask-QoS corresponding to that bid is $(\{s_i\}, \{\$100.00\})$, the price that br_k proposes for s_i in the next round must not exceed \$100.00. Otherwise, that bid will be filtered out. That bidder's other bids, if matching or exceeding corresponding ask-QoSs, can still remain in the auction.

To generate ask-QoSs, *minimum decrements* (measured by percentage) are imposed on bidders' current bids, which represent the minimum concession that bidders have to make for their bids to remain in the auction. In a large-scale scenario where the number of bids is huge, large minimum decrements are usually preferable as they efficiently filter out uncompetitive bids. On the other hand, small minimum decrements are more reasonable in small-scale scenarios because overly large minimum decrements may filter out too many (sometimes all) remaining bids, decreasing the success rate of finding a solution to the WDP. In an auction that lasts through several rounds, the number of remaining bidders usually decreases as the auction proceeds. The minimum decrement should be reduced accordingly. CASS adopts a novel mechanism, called *dynamic minimum decrement* (*DMD*) to facilitate ask-QoS generation. At the early stage of the auction, higher minimum decrements are applied to generate ask-QoSs. By doing this, DMD efficiently filters out uncompetitive bids, decreasing the number of remaining bids and reducing the complexity of the WDP. As the auction proceeds, the minimum decrement decreases to effectively exploit the QoS capacities of the remaining bidders. By doing so, DMD can improve the efficiency of the auction without sacrificing the effectiveness. The selection of DMD model (or formula) is domain-specific and is de-

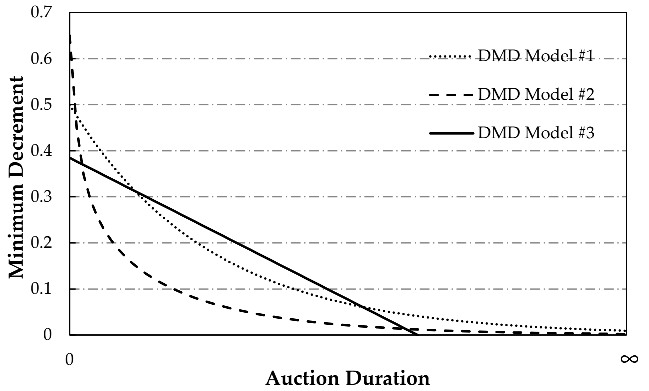


Fig. 5. Example models for DMD generation.

termined by the SBS designer. Fig. 5 illustrates three example DMD models. Apparently, the minimum decrements generated based on model #2 decreases faster over time than those based on models #1 and #3.

6.1.3 Final Deal-Sealing Round

A final deal-sealing round is a round in which bidders have a last chance to propose their bids for certain services. There are two situations where a final deal-sealing round can be held: 1) when a time limit constraint has been set for the auction and it is violated; 2) when all the bidders competing for certain services quit the auction in the same round because they cannot accept the ask-QoSs, leaving some service contracts unallocated. The result from a final deal-sealing round will determine the final winning bids for these services. If the bids proposed in the final deal-sealing round are still estimated equivalent, the final winning bids for those services will be selected randomly from the remaining bids. If the bidders still would not accept the ask-QoSs in the final deal-sealing round, the bid(s) from the last round with the highest utility would be selected as the final winning bids for those services.

6.2 Winner Determination

Based on received bids, the auctioneer needs to determine which bids are current winning bids, i.e., the winning bids in the current round. This is a multi-attribute decision making problem, which in the context of combinatorial auction is referred to as the WDP, a NP-complete problem [51].

Suppose the business process of an SBS $\mathbf{S}$ consists of r (r≥1) abstract services K={s_1, …, s_r}. Given m (m≥2) groups of bids (proposed by m bidders br_k, k=1, …, m) in the auction, each contains n received bids. Each bid b_k specifies a set of services S_k and a QoS offer Q_k. The WDP is a constraint optimisation problem (COP) that aims at selecting zero or one bid from each bid group so that the selected bids achieve the SBS designer's optimisation goal *objective*($\mathbf{S}$), while fulfilling all quality constraints for the SBS $\mathbf{S}$ C={c_1, …, c_t}.

To capture the quality constraints for $\mathbf{S}$, we first model the WDP as a constraint satisfaction problem (CSP), which consists of a finite set of variables X={X_1, …, X_n}, with respective domains D={D_1, …, D_n} listing the possible values for each variable, and a set of constraints C={c_1, …, c_t} over X. A solution to a CSP is an assignment of a value to each variable from its domain such that every constraint is satisfied. The CSP model of the WDP can be formally expressed as follows.

For m bidders, there are m×n 0-1 variables $X_{i,j}$ (i=1, …, m, j=1, …, n, and $D(X_{i,j})$={0, 1}), $X_{i,j}$ being 1 if the jth bid in the ith bid group is selected as part of the current winning bids, 0 otherwise. The constraints for the CSP model are:

XOR constraints: $$\sum_{j=1}^{n} X_{i,j} \leq 1 \quad \forall i \in [1, m] \qquad (11)$$

Coverage constraints: $$\sum_{i,j|s_k \in S_{i,j}} X_{i,j} = 1 \quad \forall k \in [1, r] \qquad (12)$$

Quality constraints: $$q_p(\mathbf{S}) < c_p \quad \forall p \in [1, t] \qquad (13)$$

where $S_{i,j}$ is the set of abstract services specified in the jth bid in the ith bid group, $q_p(\mathbf{S})$ is the pth quality parameter of $\mathbf{S}$ and can be obtained by applying the quality aggregation functions presented in Table 1.

Constraints family (11) guarantees that at most one bid is selected from each bid group, according to the XOR bidding language. Constraints family (12) guarantees full coverage of K, i.e., each abstract service is included in selected bids exactly once. Constraints family (13) ensures that all quality constraints for the SBS are fulfilled. Note that not every SBS owner has constraints for all t quality parameters of the SBS. Sometimes, there can be non-linear quality constraints due to non-linear quality aggregation. They can be approximately linearised using certain techniques [7]. However, such techniques often produce an inordinate number of variables and constraints, putting the problem beyond the practical reach of available IP tools. An overview of the tools for nonlinear IP can be found in [21]. It is still a positively active research field and the existing tools are still facing many theoretical and practical difficulties. Hence, in this research, we consider only linear quality constraints.

Solving the above CSP can generate a solution that fulfils all quality constraints for the SBS. Such a solution is called a *feasible solution*. Very often, there are many feasible solutions. Take the SBS presented in Fig. 1 for example, there might exist multiple feasible solutions that yield different overall system utility at different overall costs, as presented in Fig. 6. Now we seek to achieve the SBS owner's optimisation goal for the SBS. Given an objective function that represents the SBS owner's optimisation goal, the CSP turns into a COP. In a COP, each solution generated by solving the CSP is associated with a ranking value for the objective function. The solution with the optimal ranking value is the solution to the COP, i.e., the solution to the WDP of the auction.

Most existing approaches in the area of quality-aware service selection aims at maximising overall system quality (or utility) [2-4, 60]. However, SBS owners' optimisation goals can be various (and often conflicting to each other), e.g., to minimise the overall cost of service usage for the system or the overall response time of the system. CASS allows SBS designers to specify optimisation goals flexibly according to SBS owners' needs, which in the

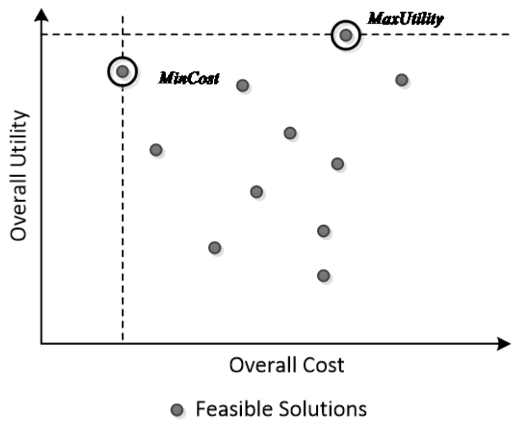


Fig. 6. Feasible and optimal solutions.

COP model are represented using different objective functions. In this research, we use the objective functions for two typical optimisation goals as examples: 1) to maximise the overall system utility; and 2) to minimise the overall cost of service usage for the SBS. A maximised overall system utility usually results in high stakeholder satisfaction as the quality constraints are exceeded to the maximum based on the utilities of available candidate services. Minimising the overall cost of service usage is a common choice for SBS owners with limited budgets as it helps them keep the overall cost within budget.

1. **Maximising the overall system utility**. The solution to this optimisation goal is the set of services that fulfills all the quality constraints and meanwhile, maximises the overall utility of the SBS, e.g., $S_{MaxUtility}$ in Fig. 6. The objective function that captures this optimisation goal is as follows:

$$objective(S): \text{maximise} \sum_{i,j|s_k \in S_{i,j}} u(s_k) \times X_{i,j} \tag{14}$$

2. **Minimising the overall cost of service usage**. The solution to this optimisation goal is the set of services that fulfills all quality constraints and meanwhile, minimises the total price of the selected services, e.g., $S_{MinCost}$ in Fig. 6. The objective function that captures this optimisation goal is as follows:

$$objective(S): \text{minimise} \sum_{i,j|s_k \in S_{i,j}} q_{price}(s_k) \times X_{i,j} \tag{15}$$

This COP can be solved by applying IP techniques [60] (or mixed integer programming technique [3, 4] if decimal variables are involved). Based on the results from solving the COP, the winning bids can be determined. If the stop criterion for the auction is met, the current winning bids become final winning bids. According to the final winning bids, the SBS designer selects the concrete services for the SBS, awards the service contracts to the winning bidders and finalises the SLAs for the concrete services. However, sometimes no feasible solutions to the COP can be found, which indicates that the current bids are not good enough to fulfil all quality constraints for the SBS. If the stop criterion (see Section 6.3) for the auction is not met, the auction will iterate, allowing the bidders to improve their bids. In such cases, we remove constraint family (13) from the COP model and find the *current best bids*, which achieve the SBS owner's optimisation goal while fulfilling constraints families (11) and (12). The auctioneer then distributes the current best bids to the bidders as the current winning bids, which help them analyse their positions in the competition - they can compare their current bids to the current winning bids and determine whether and how to improve their bids.

6.3 Stop Criteria

In Step 3 of CASS, as described in Section 5, the current winning bids are determined by solving the WDP which is the optimal solution based on the current bids. However, whether the current winning bids are the final winning bids depend on whether the auction stops. For example, the current winning bids, being the optimal solution based on the current bids, might not meet the requirements for the quality of the SBS. Thus, the auctioneer needs to determine whether the auction stops by checking if the stop criterion is met. CASS provides three stop criteria:

- **QoS fulfilment**. This stop criterion is to meet the quality constraints for the SBS. If the current winning bids fulfil or exceed the quality constraints for the SBS, the auction stops immediately and the current winning bids are considered as final winning bids. This stop criterion is used when there are specific constraints for the quality of the SBS.
- **Time limit constraint**. This stop criterion is a pre-specified time limit constraint for the auction. Similar to eBay, a timer can be set, based on the scale of the auction, to determine when the combinatorial auction is forced to stop. The timer begins to count down as the auction starts. When time is up, bidding is no longer allowed and the current winning bids or current best bids, if exist and fulfil the quality constraints for the SBS, are considered as the final winning bids. In order for the auctioneer to obtain better QoS offers for the SBS, a final deal-sealing round (see Section 6.1.3) can be performed which gives the bidders a last chance to improve their bids. If no solution is found within the time limit, the auction is considered failed. If necessary, the auction can start over. This stop criterion can prevent the auction from hanging due to potentially significant time consumption by NP-complete winner determination. It also prevents overly long auctions, especially if human intervention is involved in any parties in the auction. Thus, the time limit constraint is used with a relatively small value when time is a major concern for the auction. Otherwise, a large value can be used, allowing the auction to go on for a relatively long time, possibly days as at ebay.com.
- **Final-best-bids.** When a solution is found, it is possible that the non-winning bidders are still willing to improve their bids. The auctioneer can adopt the third stop criterion, i.e. final-best-bids, to further exploit the competition between the bidders in such situations, which is to let the auction continue until only one bidder remains for each abstract service. As the auction proceeds, bidders that cannot afford to accept the ask-QoS have to withdraw corresponding bids. Finally, only one bid remains for each abstract service and the remaining bids are considered the final winning bids. Apparently, this stop criterion is appropriate when the quality of the SBS is the major concern and the time taken by the auction process is not.

6.4 Ask-QoS Generation

In Step 3 of CASS, as described in Section 5, the auctioneer also checks whether the stop criterion is met. If not, the auction needs to continue by starting a new round. In Step 4, CASS provides bidders with non-linear and non-anonymous ask-QoSs to guide their bidding and filters out uncompetitive bids. Bidders that cannot afford the ask-QoSs have to withdraw corresponding bids. Utilising ask-QoS, the auctioneer can make sure that the auction proceeds in the right direction towards a satisfactory solution by demanding better bids from the bidders until their QoS capacities are exhausted or the auction stops. In addition, by filtering out uncompetitive bids, the application of ask-QoSs can reduce the complexity of solving the WDP because the size of the search space (i.e., number of bids) is reduced.

Non-linear and non-anonymous ask-QoSs are defined as follows:

Definition 4: (Non-Linear Ask-QoS). Given ask-QoSs $(\{s_m\}, \{AQ_i(s_m)\})$, $(\{s_n\}, \{AQ_j(s_n)\})$, $(\{s_m, s_n\}, \{AQ_k(s_m), AQ_k(s_n)\})$ that involve two abstract services s_m and s_n ($s_m \neq s_n$), non-linear ask-QoSs allow:

$$aq_{i,p}(s_m)+aq_{j,p}(s_n)\neq aq_{k,p}(s_m)+aq_{k,p}(s_n) \tag{16}$$

where $aq_{i,p}(s_m)$, $aq_{j,p}(s_n)$, $aq_{k,p}(s_m)$ and $aq_{k,p}(s_n)$ represent the asked offers for the pth QoS parameter of services s_m and s_n specified in $AQ_i(s_m)$, $AQ_j(s_n)$, $AQ_k(s_m)$ and $AQ_k(s_n)$.

For example, a bidder can be asked for the price of \$200.00 for service s_m, \$300.00 for service s_n and \$470.00 for the combination of s_m and s_n.

Definition 5: (Non-anonymous Ask-QoS). Given two ask-QoSs $(\{s_m\}, \{AQ_j(s_m)\})$ and $(\{s_m\}, \{AQ_k(s_m)\})$, for two different bidders, non-anonymous ask-QoSs allow:

$$aq_{j,p}(s_m)\neq aq_{k,p}(s_m) \tag{17}$$

where $aq_{j,p}(s_m)$ and $aq_{k,p}(s_m)$ represent the pth QoS parameter of s_m specified in the $AQ_j(s_m)$ and $AQ_k(s_m)$.

For example, the auctioneer can ask a bidder for the price of \$200.00 for an abstract service while asking another bidder for the price of \$250.00 for the same abstract service.

Non-linear ask-QoS allows CASS to take into account the complementarities between services. Non-anonymous ask-QoS allows CASS to elicit bidders' dissimilar QoS capacities. For instance, a bidder with potential capacity for providing a low price can be asked for a lower price while another bidder capable of completing the task fast is asked for a short response time.

We compute the ask-QoS by analysing service providers' historical bids. Here, we introduce a novel concept called *Surplus Bidding Space*, which models the bidders' potential capacity for improving their offers for a specific QoS parameter. Given the current offer for the pth QoS parameter of service s_i proposed by bidder br_k in the current round, denoted by $q_p^{k,cur}(s_i)$, and a series of br_k's historical offers for $q_p(s_i)$, denoted by $q_p^{k,1}(s_i)$, …, $q_p^{k,n}(s_i)$, we first calculate the standard score of $q_p(s_i)$ to reflect how far $q_p^{k,cur}(s_i)$ falls above or below the average value of $q_p^{k,1}(s_i)$, …, $q_p^{k,n}(s_i)$:

$$z(q_p^{k,cur}(s_i)) = (q_p^{k,cur}(s_i) - q_p^{k,ave}(s_i)) / \sigma \tag{18}$$

where $q_p^{k,ave}(s_i)$ and σ are the average value and standard deviation of $q_p^{k,cur}(s_i)$, $q_p^{k,1}(s_i)$, …, $q_p^{k,n}(s_i)$.

Then, we calculate the surplus bidding space of br_k for $q_p(s_i)$ by applying the Simple Additive Weighting (SAW) technique [57]:

$$\varepsilon_p^k(s_i) = \begin{cases} \dfrac{z(q_p^{k,cur}(s_i)) - z(q_p^{k,min}(s_i))}{z(q_p^{k,max}(s_i)) - z(q_p^{k,min}(s_i))} & \text{if } z(q_p^{k,max}(s_i)) \neq z(q_p^{k,min}(s_i)) \\ 1 & \text{if } z(q_p^{k,max}(s_i)) = z(q_p^{k,min}(s_i)) \end{cases} \tag{19}$$

where $\varepsilon_p^k(s_i) \in [0,\ 1]$, $z(q_p^{k,max}(s_i))$ and $z(q_p^{k,min}(s_i))$ represent the standard scores of the maximum and minimum offer for the pth QoS parameter of service s_i that br_k has ever provided.

Surplus bidding space indicates the potential capacity of a bidder for a better QoS offer compared to its current offer. The minimum decrement (or minimum increment in the case of positive QoS parameter) imposed by an ask-QoS on a bidder is dependent on the surplus bidding space of the bidder. For example, a bidder with a high $\varepsilon_p^k(s_i)$ can be pushed hard on $q_p(s_i)$ with an ask-QoS that imposes a large minimum decrement on the bidder's current QoS offer for $q_p(s_i)$. For an unknown bidder, whose historical bids are unavailable, formulas (18) and (19) cannot be applied to calculate the bidder's surplus bidding space. In such cases, the average value of the surplus bidding space of other bidders competing for the same abstract service will be used to generate the ask-QoS for the unknown bidder.

As described in Section 6.1.2, high minimum decrements can speed up the auction process at the early stage of the auction by efficiently filtering out uncompetitive bids, while low minimum decrements can maximise system optimality by effectively exhausting the surplus bidding space of the remaining bidders at the late stage. Ask-QoS generation in CASS is based on DMD, aiming at guaranteeing both the effectiveness and efficiency of the auction. In general, there are two factors that affect the determination of DMD: the surplus bidding space of the bidders and the adopted DMD model. According to Fig. 7, the generation of the minimum decrement in the pth QoS parameter of service s_i for bidder br_k proceeds as follows:

- Calculate the surplus bidding space $\varepsilon_p^k(s_i)$ of br_k for the pth QoS parameter of service s_i;
- Draw a straight line from the origin with $\varepsilon_p^k(s_i)$ as the slope;
- Locate the point at which the straight line intersects with the DMD model curve.

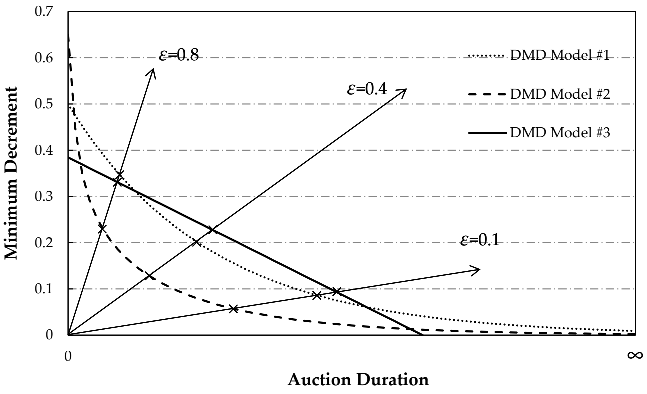


Fig. 7. Generation of dynamic minimum decrement (DMD).

- Obtain the y axis value of the intersection point and that is the minimum decrement in the pth QoS parameter of service s_i for br_k.

As the auction proceeds, the surplus bidding space of a bidder shrinks. Accordingly, the slope of the solid straight line in Fig. 7 becomes smaller. Then solid straight line intersects with the DMD model curve at a point with a lower y axis value. In this way, the minimum decrement applied to generate the ask-QoS for the bidder decreases over time according to its surplus bidding space and the adopted DMD model as the auction proceeds.

According to formulas (18) and (19), when $q_p^{k,cur}(s_i) < q_p^{k,min}(s_i)$, $\varepsilon_p^k(s_i) <0$. According to Fig. 7, the solid straight line with $\varepsilon_p^k(s_i)$ as the slope does not intersect with the curve of the DMD model. In such cases, the value of $\varepsilon_p^k(s_i)$ is invalid. It happens when the QoS offer specified in the current bid is out of the range of the service provider's historical bids, which can be caused by the bidder drastically changing its bidding behaviour. Thus, using the bidder's historical bids to estimate its surplus bidding space becomes infeasible. In such cases, if that bid is part of the current winning bids, it is allowed to remain in the auction until beaten by other bids. Otherwise, half of the DMD value in the previous round will be used as default value to generate the ask-QoS for the bidder in the current round.

7 Experimental Evaluation

This section presents the experimental evaluation of CASS, focusing on the comparison with existing optimisation approaches in terms of effectiveness (measured by the success rate of finding an optimisation solution and the system optimality) and efficiency (measured by auction duration and the coordination overhead). It ends with a validity analysis.

7.1 Prototype Implementation

We have implemented a prototype of CASS in Java using JDK 1.6.0 and Eclipse Java IDE. It implements the mechanisms introduced before. Given as input the functional specification of the business process of an SBS, a set of quality constraints for the SBS and an optimisation objective, an iterative combinatorial auction can be held to realise SLA negotiation for the SBS. For solving the COP (i.e., the WDP) introduced in Section 6.2, the prototype uses IBM CPLEX v12.2, a linear programming solver.

7.2 Experimental Setup

To evaluate CASS, we have conducted a series of in-lab experiments. The evaluation process mimicked multiple SBSs. In the experiments, we used the prototype to hold combinatorial auctions with simulated bidders. A random proportion (between 40% and 60%) of those bidders were given the ability to bid for randomly generated combinations of services as in reality not every bidder has the ability or is willing to bid for multiple services. Those bidders applied randomly generated discounts (5%-15%) to their QoS offers when bidding for combinations of services. Theoretically, the maximum number of different combinations of services that a bidder can bid for in an auction with m abstract services to be auctioned is $\sum_{i=1}^{m} C_m^i$. To model the bidders' behaviours realistically, we restricted this number to be equivalent to the number of abstract services in the experiments. For example, if there are 100 abstract services of the SBS, each bidder can bid for up to 100 different combinations of services. In real world, it is the bidders' own decisions to bid for how many and which services or combinations of services. CASS itself does not limit the way that bidders combine services.

Bidders' bidding behaviors can be very different from each other. Even a same bidder can adopt different bidding strategies when bidding for different items, presenting dissimilar bidding behaviours. Hence, we did not adopt specific probability distributions in generating service providers' historical bids in the experiments. Instead, the historical bids of the bidders were generated based on the QoS information provided in QWS – a publicly available Web service dataset that comprises measurements of nine QoS parameters of over 2500 real-world Web services [1]. We randomly selected the quality of the services in QWS to represent bidders' historical bids for specific abstract services. For each abstract service, a bidder had at least five historical bids. The QoS offers specified in a bidder's first bid proposed in the first round were generated based on its historical bids. For example, if a bidder's best historical offer for the price of a service is \$2,000, the price proposed in its first bid was selected from a randomly generated interval, e.g., [\$1,500, \$2,500]. This configuration captured the nature of the bidders with various surplus bidding spaces. The optimisation goal in the experiments is to maximise the overall system utility (see Section 6.2). When severe quality constraints are imposed on the SBS, e.g., limited budget and stringent response time, it is possible that no solution can be found that meet these quality constraints [4]. Allowing bidders to improve their QoS offers and exploiting competition between bidders is a promising attempt to address this issue. To evaluate the success rate of CASS in different situations, we simulated three types of scenarios where different difficulty levels of quality constraints were imposed on the SBS:

- **Simple**. This type of quality constraint set is relatively easy to be satisfied on all quality parameters.
- **Medium**. This type of quality constraint set is more difficult to be satisfied than the "simple" level as the constraints imposed on some quality parameters of the SBS are demanding.
- **Severe**. This type of quality constraint set is the most difficult to be satisfied as the constraints imposed on all quality parameters of the SBS are demanding.

Fig. 8 presents randomly generated quality constraint sets on different difficulty levels for an abstract service s_i. In Fig. 8, $q_{rt}^{min}(s_i)$ and $q_{rt}^{max}(s_i)$ are the minimum and maximum values of all bidders' historical offers for the response time of s_i. $q_{price}^{min}(s_i)$ and $q_{price}^{max}(s_i)$ are the minimum and maximum values of all bidders' historical offers for the price of s_i.

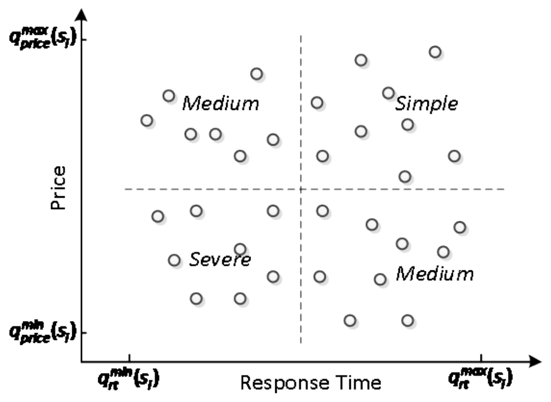


Fig. 8. Quality constraints on different difficulty levels.

The quality constraints for the SBS were generated by aggregating randomly generated quality constraints for individual abstract services of the SBS. For example, for an SBS $\mathbf{S}=\{s_1, \ldots, s_n\}$, given the set of randomly generated constraints for the price of s_i (i=1, …, n) denoted by $c_{price}(s_i)$ (i=1, …, n), the constraint for the price of $\mathbf{S}$ is $\sum_{i=1}^{n} c_{price}(s_i)$.

When a solution could not be found, ask-QoSs were generated based on bidders' historical bids and the bidders would improve their bids according to the ask-QoSs to start a new round. In the experiments, the bidders would continue to improve their bids until their surplus bidding space are exhausted. For ask-QoS computation, we adopted DMD model #3 presented in Fig. 5 unless particularly specified. For the stop criterion, we selected *QoS fulfillment* (see Section 6.3) as it is also the criterion adopted by other similar work [2-4, 60] to determine a solution.

There are five factors that influence the effectiveness and efficiency of CASS: 1) the number of quality constraints imposed on the SBS; 2) the number of abstract services included in the business process of the SBS; 3) the number of bidders participating in the auction; 4) the difficulty level of the quality constraints imposed on the SBS; and 5) the number of bids proposed by each bidder. We have conducted four sets of experiments. The configuration of each set of experiment is presented in Table 2. In set #1, we increased the number of quality constraints from 1 to 10 in steps of 1 while fixing the number of abstract services at 10 and the number of bidders at 100. In set #2, we increased the number of abstract services from 10 to 100 while fixing the number of quality constraints at 2 and the number of bidders at 100. In set #3, we increased the number of bidders from 100 to 1000 while fixing the number of quality constraints at 2 and the number of abstract services at 10. In set #4, we increased the overall scale – the numbers of quality parameters, services and bidders combined. We then changed the difficulty level of quality constraints, creating three subsets of experiments in each set of experiments, i.e., *simple*, *medium* and *severe*. In each subset of experiment, 100 instances of experiments were run and the collected results were averaged. The maximum number of different combinations of services that a bidder can bid is equivalent to the number of abstract services. Increase in this number would increase the total number of bids and scale up the experiment. However, the maximum of the total number of bids, given the current experimental configuration, was already 60,000, which we believe was large enough for the purpose of the experiment. Hence, we did not change the maximum number of different combinations of services that a bidder can bid for in the experiment.

The experiments were conducted on a machine with AMD Athlon(tm) X4 640 3.00GHz CPU and 8 GB RAM, running Windows 7 x64 Ultimate.

7.3 Effectiveness Evaluation

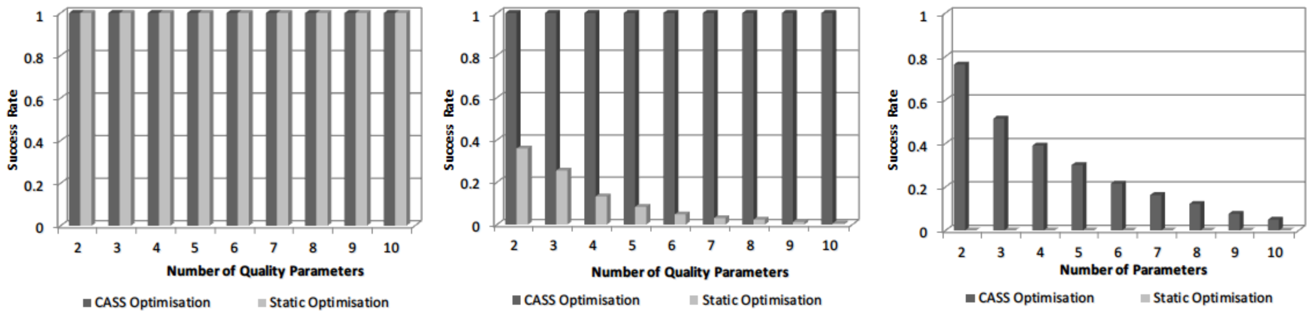
To evaluate the effectiveness of CASS, we implemented the optimisation approach adopted in [2-4, 60] for comparison, which is also based on IP. We refer to this optimisation approach as *Static Optimisation* as it does not consider the complementarities between services and the competition among service providers. In the experiments for Static Optimisation, the bidders only bid for individual services.

We first compare the effectiveness of *CASS Optimisation* (i.e., optimisation based on CASS) and Static Optimisation by their success rates – the percentage of scenarios where a solution could be found that meets the quality constraints for the SBS, and meanwhile, achieves the optimisation goal.

TABLE 2
EXPERIMENTAL CONFIGURATION

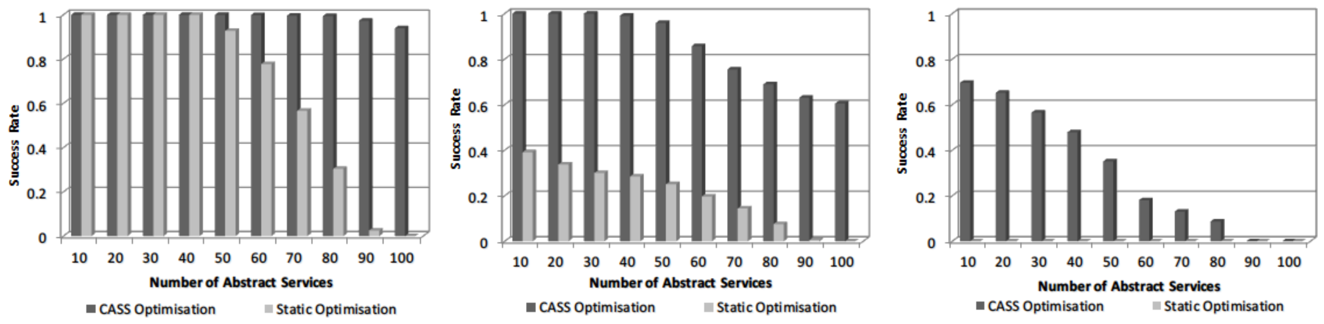
Factor	**Experiment Set #1**	**Experiment Set #2**	**Experiment Set #3**	**Experiment Set #4**
Number of Quality Constraints	From 1 to 10 in step of 1	2	2	From 1 to 10 in steps of 1
Number of Abstract Services	10	From 10 to 100 in steps of 10	10	From 10 to 100 in steps of 10
Number of Bidders	100	100	From 100 to 1000 in steps of 100	From 100 to 1000 in steps of 100

Fig. 9 - 12 present the results obtained from the experiments. As illustrated, CASS Optimisation significantly outperforms Static Optimisation, 81.11% versus 38.15% on average across all experiments. As the scenario scales up, the success rate of Static Optimisation drops quickly in most cases while CASS Optimisation maintains significantly higher success rate in those cases compared to Static Optimisation. In particular, in the "severe" subsets of experiments, the success rate of Static Optimisation remains zero despite the numbers of quality parameters, services and bidders while CASS Optimisation was still able to solve certain percentages of those "severe" cases. Specifically, the average success rates of CASS Optimisation are 28.82% 31.40%, 89.85% and 39.36% in the "se-



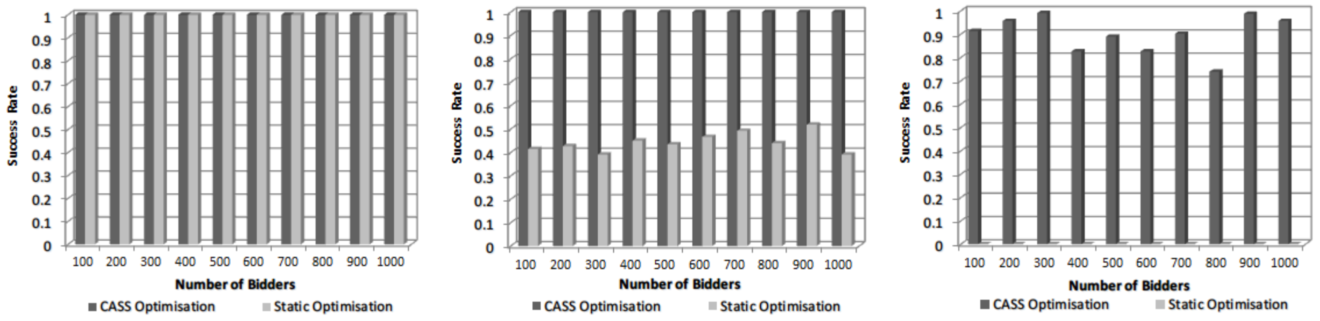
(a) Simple (b) Medium (c) Severe

Fig. 9. Set #1: success rate vs. number of quality constraints.



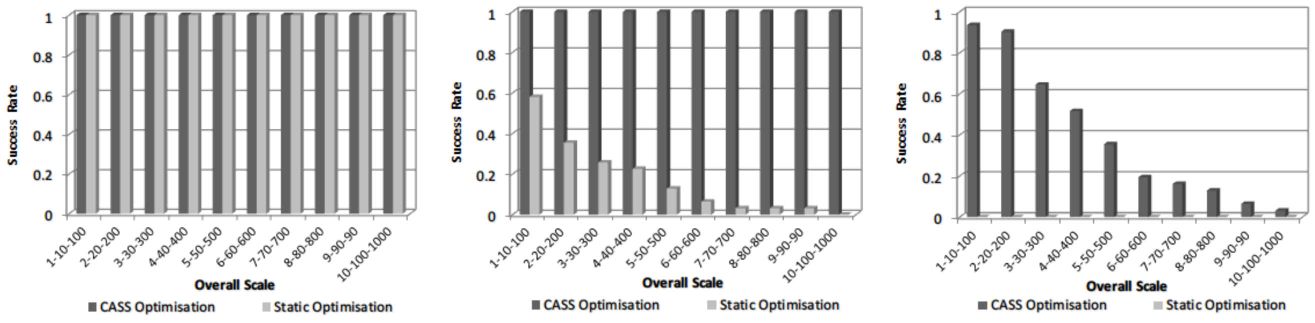
(a) Simple (b) Medium (c) Severe

Fig. 10. Set #2: success rate vs. number of abstract services.



(a) Simple (b) Medium (c) Severe

Fig. 11. Set #3: success rate vs. number of bidders.



(a) Simple (b) Medium (c) Severe

Fig. 12. Set #4: success rate vs. overall scale.

vere" scenarios of experiment sets #1, #2, #3 and #4 respectively, 47.36% on average.

As illustrated by Fig. 9 (a) and (b), when not all the quality constraints were demanding, CASS Optimisation always found a solution. As shown in Fig. 9 (c), the success rate of CASS optimisation decreases from 76% to 5% as the number of quality constraints increases from 2 to 10. The reason is that the increase in the number of quality constraints made it increasingly difficult to satisfy all "severe" quality constraints imposed on the SBS. On average, the success rate of CASS optimisation across all scenarios in this set of experiments is over twice that of Static Optimisation, i.e., 76.27% vs. 36.86%. Based on a total of 2,700 samples collected in this series of experiments, we conducted chi-square test with the null hypothesis that CASS has no difference to Static Optimisation in finding a solution as the number of quality constraints increases. Based on the samples, the test statistic value x^2 is 853.26, higher than the critical value of the test statistic, which is 7.88 (confidence level=0.005), for p=1E-187. Thus, we can reject the null hypothesis and conclude that CASS is much more effective than Static Optimisation in finding a solution as the number of quality constraints increases.

The results observed in Fig. 10 are similar to those in Fig. 9. In general, the success rate decreases as the number of abstract services increases. Compared to the number of quality constraints, the increase in the number of abstract services influences the success rate more significantly. In the cases of 9 and 10 services, even CASS Optimisation failed to find a solution. However, CASS Optimisation still beats Static Optimisation by a considerably large margin in success rate in this set of experiments. On average, the success rate of CASS optimisation across all scenarios in this set of experiments is approximately 71.76% and number for Static Optimisation is only 28.62%. Based on the samples collected in this series of experiments, the test statistic value x^2 by chi-square test is 1003.64, higher than the critical value of the test statistic, which is 7.88 (confidence level=0.005), for p=3E-220. Thus, we conclude that CASS is much more effective than Static Optimisation in finding a solution as the number of abstract services increases.

The results presented in Fig. 11 illustrate that the increase in the number of bidders does not directly influence the success rate. In this set of experiments, the difficulty level of the quality constraints is the only factor that affects the success rate. Again, CASS Optimisation shows a significant advantage over Static Optimisation with a success rate margin of 48.53% (96.62% vs. 48.09%). In particular, when the difficulty level of quality constraints is "severe", CASS obtains much higher success rates (89.85%) in this set of experiments than in the previous two sets (28.82% for set #1 and 31.40% for set #2). The reason is that compared to the previous two sets of experiments, there were more bidders in the auction in set #3, which led to fiercer competition among bidders. By exploiting that competition, CASS manages to find a solution most of the times even when the quality constraints for the SBS are severe. Based on the samples collected in this series of experiments, the test statistic value x^2 by chi-square test is 1591.09, higher than the critical value of the test statistic, which is 7.88 (confidence level=0.005), for p=0. Thus, we conclude that CASS is much more effective than Static Optimisation in finding a solution as the number of bidders increases.

In set #4, the intensity of the competition among the bidders remained at the same level – the average number of bidders competing for each service was 10. However, the increase in the number of quality constraints made it more difficult to find a solution. As illustrated by Fig. 12, the average success rates obtained by both CASS optimisation and Static Optimisation in the "medium" and "severe" experiments decreased as the experiments scaled up. Compared to Static Optimisation, CASS Optimisation obtained significantly higher success rates. In particular, while Static Optimisation failed completely in all scenarios in the "severe" scenarios, CASS Optimisation obtained an average success rate of 39.36%. Based on the samples collected in this series of experiments, the test statistic value x^2 by chi-square test is 929.19, higher than the critical value of the test statistic, which is 7.88 (confidence level=0.005), for p=4E-204. Thus, we conclude that CASS is much more effective than Static Optimisation in finding a solution as the overall scale increases.

In order to statistically compare our approach and Static Optimisation in finding a solution in general, we also performed a chi-square test based on all the samples collected in the abovementioned four series of experiments. The test statistic value x^2 is 4140.54, higher than the critical value of the test statistic, which is 7.88 (confidence level=0.005), for p=0. Based on the results of the chi-square test, we conclude that CASS is much more effective than Static Optimisation in finding a solution.

In order to find a satisfactory solution, the CASS Optimisation method and Static Optimisation method both seek to achieve the SBS owner's system optimisation goal – to maximise the overall system utility. Hence, higher overall system utility indicates higher system optimality, and as a result, higher success rate of finding a solution. In order to compare the two optimisation methods in system optimality, we averaged the overall system utility obtained (when a solution was found) by CASS Optimisation and Static Optimisation in different experimental scenarios. The results are presented in Table 3. Based on the results, we performed Wilcoxon test, where the null hypothesis H_0 was that there is no difference between the system utility obtained by Static Optimisation and CASS. The test statistic value p by the Wilcoxon test is 0.002, much lower than 0.05. So we reject the null hypothesis. Thus, the results clearly demonstrate the advantage of CASS over Static Optimisation in obtaining system utility. Furthermore, for an intuitive comparision, we also calculated u_{CASS}/u_{Static}. The results are presented in Table 4, where "N/A" represents that the comparison is not applicable in that case because Static Optimisation could not find a solution. As shown, CASS Optimisation outperforms Static Optimisation by margins of between 23% and 66%. Across all experiements in the "simple" and "medium" scenarios, the overall system

TABLE 3
SYSTEM UTILITY COMPARISON (*U_{CASS}*/*U_{STATIC}*)

Experiment Set	Difficulty Level of Quality Constraints	Static Optimisation	CASS
Set #1	Easy	3.34	4.82
	Medium	3.24	3.97
	Severe	0	3.23
Set #2	Easy	0.95	1.58
	Medium	0.97	1.48
	Severe	0	0.99
Set #3	Easy	1.31	1.74
	Medium	1.34	1.70
	Severe	0	1.26
Set #4	Easy	2.97	4.347
	Medium	2.7	3.44
	Severe	0	2.79

TABLE 4
SYSTEM OPTIMALITY (U_{CASS}/U_{STATIC})

Changing Factor	Difficulty Level of Quality Constraints		
	Simple	Medium	Severe
Number of Quality Constraints	1.44	1.23	N/A
Number of Abstract Services	1.66	1.53	N/A
Number of Bidders	1.32	1.27	N/A
Overall Scale	1.59	1.27	N/A

utility obtained by CASS Optimisation is approximately 1.41 times the overall system utility obtained by Static Optimisation.

The experimental results indicate that compared to existing static optimisation approaches, CASS Optimisation significantly increases the possibility of finding solutions for SBSs and improves the system optimality by exploiting the complementarities between services and the competition among service providers.

7.4 Efficiency Evaluation

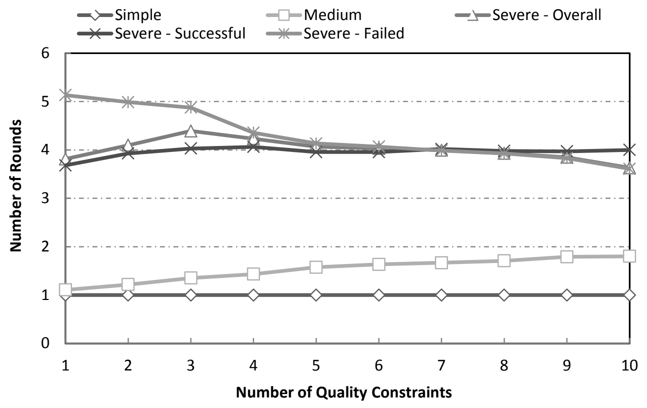
A CASS auction may last multiple rounds. In each round, the auctioneer needs to coordinate the auction by 1) determining the winners for each round; and 2) generating ask-QoSs for bidders. Thus, there are two aspects to the efficiency of CASS: *auction duration* and *coordination overhead*.

Auction duration, measured by the number of auction rounds, is a relevant efficiency concern of CASS because long auction duration implies low efficiency and high coordination overhead – the WDP needs to be solved and ask-QoSs be generated a number of times during the auction. In addition, increased auction duration may reduce auctioneer's profit from the auction because a certain amount of administrative cost for maintaining the auction applies in each round of the auction. When the auction proceeds to later stages, the marginal benefit for the auctioneer from the bidders' competition may decrease dramatically and, in the worst-case scenario, is not even worth the additional administrative cost. However, a longer auction usually has more potentials to achieve the SBS owner's optimisation goal because it gives the auctioneer the necessary time to elicit high-value bidders' surplus bidding space better. Apparently, there is a tradeoff between the effectiveness and the efficiency of the auction.

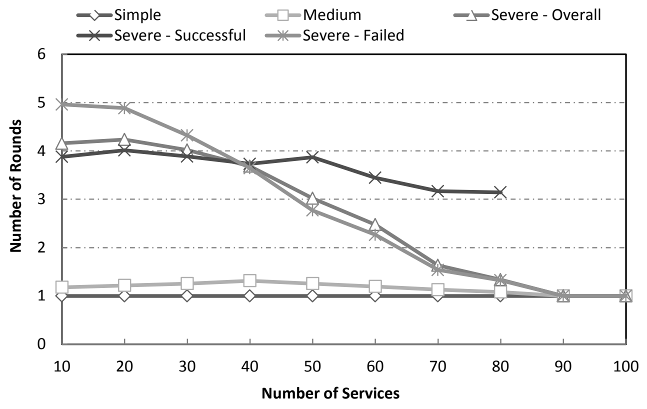
Fig. 13 presents the average number of rounds taken by the auction to complete in the experiments. The results include successful (where a solution is found) and failed auctions (where a solution cannot be found). Generally speaking, the auction duration increases with the difficulty level of quality constraints. Across all "simple" subsets of experiments, CASS takes exactly one round to complete. In the case of "medium", the average auction duration is between one and two rounds. Specifically, the average auction duration is 1.53 rounds in experiment set #1, 1.19 rounds in set #2, 1.19 rounds in set #3 and 1.55 rounds in set #4. In the "severe" scenarios, CASS takes between three and five rounds to complete, 4.00 rounds on average in set #1, 2.66 rounds in set #2, 3.88 rounds in set #3 and 4.00 rounds in set #4.

In set #2, as depicted in Fig. 13 (b), when the number of abstract services exceeded 80, CASS could not find a solution. CASS is very efficient in those scenarios – it always took only one round for CASS to find out that a solution could not be found.

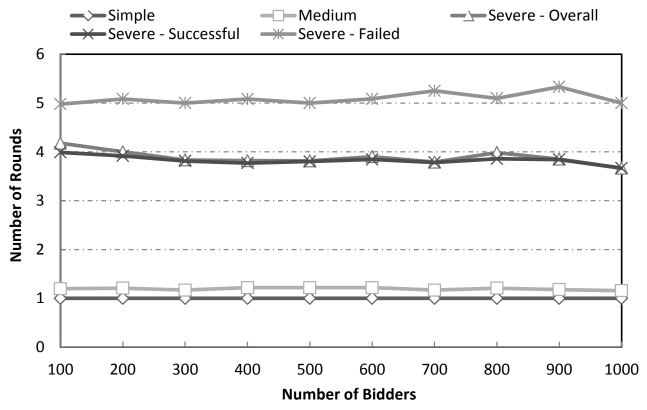
The success rate significantly influences the average auction duration. As presented in Section 7.3, as the experimental scenario scales up, the success rate decreases most of the time – it gets harder for CASS to find a solution. However, it gets easier for CASS to find out that a solution cannot be found. The influence of success rate on average auction duration is particularly significant in the "severe" scenarios where the success rate drops relatively quickly as the scenario scales up. As shown in Fig. 13, in the "severe" scenarios, when the scenario scale begins to increase, although requires increasingly more rounds to find a solution, CASS can still achieve relatively high success rate. As the scenario scale continues to increase, the success rate begins to decrease and the average duration of successful auctions contributes less and less to the average overall auction duration. However, taking less rounds to complete, failed auctions become more dominant in the determination of the average overall auction duration. As reflected in Fig. 13, the average auction duration (overall) shows an increasing and then a decreasing trend as the scenario scales up.



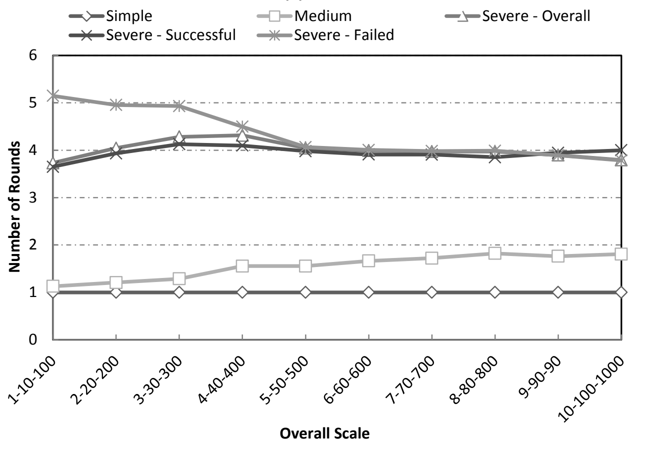
(a) Set #1



(b) Set #2



(c) Set #3



(d) Set #4 (number of quality constraints - number of abstract services - number of bidders)

Fig. 13. Auction duration vs. increasing scale.

The auction duration may vary in experiments configured in a different way or in real-world scenarios. It largely depends on the difficulty level of the quality constraints imposed on the SBSs. For example, a set of “severe” quality constraints usually requires that the bidders improve their bids many times so that a solution can be found. In both in-lab experiments and real-world scenarios, the auction duration is determined by how soon the surplus bidding spaces of the bidders (especially the high-value ones) are exhausted. In both successful and failed auctions, the efficiency of CASS in terms of auction duration is guaranteed by the application of DMD-based ask-QoS (see Section 6.4). As introduced in Section 6.4, DMD-based ask-QoS ensures that the auction proceeds fast in the right direction towards a solution. The experimental results prove the effectiveness of ask-QoS in coordinating bidders’ bidding behaviours.

Given that ask-QoS in CASS is generated based on DMD, the adopted DMD model is an important factor that impacts the duration of CASS. To demonstrate the impact of DMD models on auction duration, we conducted another set of experiments where DMD model #2 in Fig. 5 was adopted for ask-QoS generation. The results are presented in Table 5. As shown, across the “simple” subsets of experiments, it always took only one round for CASS to complete. In such cases, the ask-QoSs were not generated. Thus, it did not make a difference whether to adopt DMD model #2 or #3 for ask-QoS generation. In the “medium” experiments, CASS took no more than two rounds to complete. Accordingly, ask-QoS only needs to be generated only once at most. Thus, the difference between DMD models #2 and #3 in both success rate and auction duration is not significant. In the “severe” experiments, the adoption of DMD model #2 led to lower DMDs compared to DMD model #3. Thus, the ask-QoSs filtered out the uncompetitive bids slower when DMD

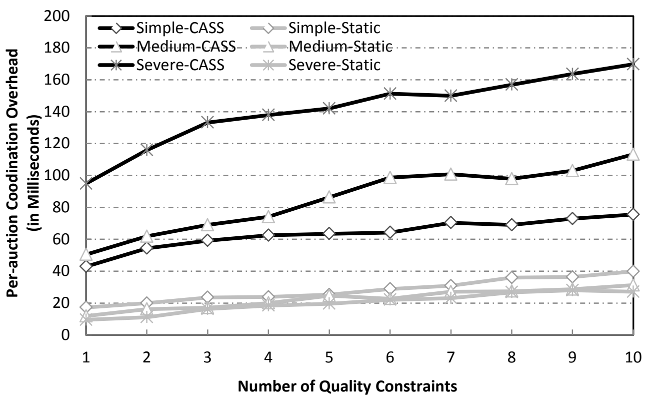
TABLE 5
DMD MODEL #2 VS. MODEL #3
(AVERAGE AUCTION DURATION/SUCCESS RATE)

Changing Factor	**Difficulty Level of Quality Constraints**					
	Simple		**Medium**		**Severe**	
	DMD Model #2	DMD Model #3	DMD Model #2	DMD Model #3	DMD Model #2	DMD Model #3
Number of Quality Constraints	1.00 / 1.00	1.00 / 1.00	1.53 / 1.00	1.53 / 1.00	5.85 / 0.38	4.00 / 0.29
Number of Abstract Services	1.00 / 1.00	1.00 / 1.00	1.16 / 0.81	1.19 / 0.85	3.74 / 0.37	2.66 / 0.31
Number of Bidders	1.00 / 1.00	1.00 / 1.00	1.21 / 1.00	1.19 / 1.00	5.48 / 0.97	3.88 / 0.90
Overall Scale	1.00 / 1.00	1.00 / 1.00	1.54 / 1.00	1.55 / 1.00	5.81 / 0.44	4.00 / 0.39

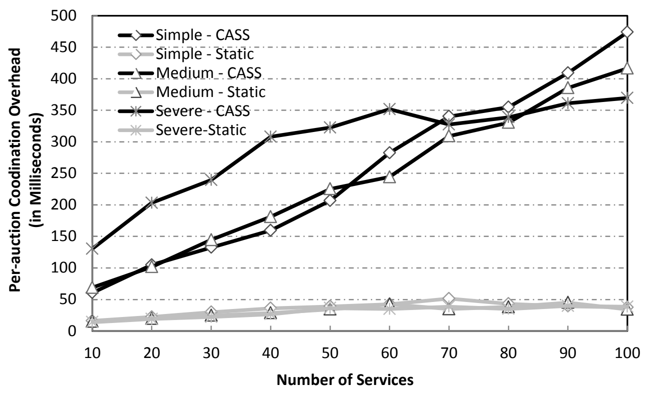
model #2 was adopted for ask-QoS generation. Thus, it took more rounds for CASS to complete when DMD model #2 was adopted. However, as said in Section 6.1.2, large DMDs may decrease the success rate of finding a solution because they sometimes filter out too many (sometimes all) remaining bids. This is confirmed by the results presented in Table 5 – in the “severe” experiments the average success rates obtained by CASS from longer auctions were higher than those obtained from shorter ones.Besides the number of auction rounds, the coordination overhead is another very important efficiency measurement. When the scenario scale is large, solving the NP-complete WDPs and generating ask-QoSs for bidders can take a significant amount of time. In order to evaluate the coordination overhead of CASS, we calculated the total CPU time (measured in milliseconds) taken per auction in winner determination and ask-QoS generation. The results are shown in Fig. 14 in comparison with Static Optimisation which only needs to solve the WDP once in each instance of experiment. From the results, we observe that the coordination overhead of CASS is significantly larger than that of Static Optimisation. The reasons are twofold. First, CASS sometimes needs more than one round to complete, requiring the WDP to be solved and ask-QoS to be generated for multiple times during an auction. Another major reason is that the search space of the WDP in CASS Optimisation is considerably larger than that in Static Optimisation. In the experiments for CASS Optimisation, 40% to 60% bidders were allowed to bid for combinations of services, creating remarkably more bids than Static Optimisation. For example, in an experiment with 1000 bidders and 100 services, the maximum number of bids created in CASS is 60,000 while the number for Static Optimisation is only 1,000. Thus, solving the WDP in CASS is much more complicated and time consuming than that in Static Optimisation. In Fig. 14, we also observe that the per-auction coordination overhead of CASS is insignificant when the scale of experimental scenario is relatively small. As the experimental scenario scales up, the per-auction coordination overhead, although increases significantly, especially in experiment set #4, is not very large. In the worst-case scenario, i.e., the “severe” scenarios in experiment set #4, the per-auction coordination in the largest-scale scenarios, being the highest across all scenarios in the experiments, is only 16 seconds.

In general, we believe that the coordination overhead of CASS is not negligible but satisfactory. In experiment sets #1, #2 and #3, the average per-auction coordination overhead is no more than 1.5 seconds. In experiment set #4, the per-auction coordination overhead increases significantly, approaching approximately 16 seconds as the scenario scales up to 1000 bidders, 100 services and 10 quality constraints. As presented before, in such scenarios, the average auction duration is around four rounds. Thus, the average per-round coordination overhead is four seconds approximately.

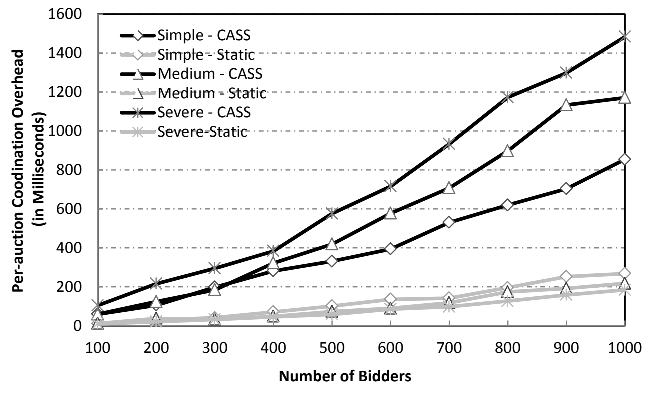
According to the results presented before, CASS needs more rounds to complete as the difficulty level of quality constraints increases. However, as demonstrated by Fig. 14, the corresponding increase in per-auction coordina-



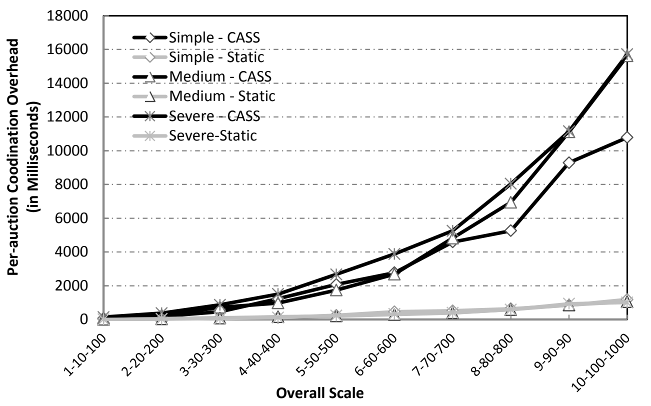
(a) Set #1



(b) Set #2



(c) Set #3



(d) Set #4

Fig. 14. Per auction coordination overhead vs. increasing scale.

tion overhead is insignificant. In the early rounds across all the auctions, e.g., the first and second rounds, the total numbers of bids were large and solving the WDPs took a relatively large amount of time. As the auction proceeded, the total numbers of bids decreased significantly because the adoption of ask-QoS filtered out a lot of uncompetitive bids each round. Solving the WDPs became much easier than in the earlier rounds. In addition, as uncompetitive bids were filtered out, the number of bidders remaining in the auction also decreased quickly. Generating ask-QoSs for the remaining bidders were also much less time consuming than in the earlier rounds. This demonstrates that severe quality constraints, although increase the auction duration, do not significantly impact the per-auction coordination overhead.

The experimental results demonstrate that CASS often requires more than one round to complete when the quality constraints for the SBS are not easy to satisfy. In addition, coordination overhead applies in CASS when the auction needs to iterate. However, given the significant increases in success rate and system optimality, we believe that the efficiency of CASS is satisfactory in most real-world applications in terms of both auction duration and coordination overhead.

7.5 Threats to Validity

Here we discuss the threats to the validity of the evaluation on our approach.

Threats to construct validity. The main threat to the construct validity of our evaluation is the comparison with Static Optimisation. Static Optimisation, based on IP, is one of the most popular approaches to QoS-aware service composition [3, 4, 60], and thus is used as a baseline for comparison in our evaluation. In our approach, the quality of the candidate services can be improved. However, in the Static Optimisation, the quality of the candidate services is static. As a result, better results, i.e., higher success rate and system utility, tend to be obtainable by our approach. Thus, the main threat to construct validity is that whether the comparison with Static Optimisation can properly demonstrate the effectiveness of our approach in finding a solution to SBS, especially in scenarios where the quality constraints for SBSs are stringent. To minimise this threat, we changed different configuration factors (as shown in Table 2) to simulate scenarios with different stringency of quality constraints for SBSs. By doing so, we could evaluate our approach by not only the comparison with Static Optimisation, by also the demonstration of how the change in the stringency of quality constraints impacts on the success rate obtained by our approach.

Threats to external validity. The main threat to the external validity of our evaluation is the representativeness of the bids simulated in the experiments. There are two aspects. The first one is that we generated bidders' historical bids based on quality information randomly selected from the QWS dataset [1], a dataset widely used in research on QoS-aware service composition. In an auction, a bid represents the bidder's offer. In a service composition scenario, it includes the QoS that the service provider is willing to offer. Once the offer is accepted and an SLA is contracted, the QoS specified in the offer becomes a bottom line for the real service provision, i.e., the QoS perceived by the users at runtime must not be worse than the QoS specified in the offer. For the service providers' perspective, it is usually to their best interests to maintain the QoS at or just above the promised levels as it requires the minimal service resources to keep their promises. Thus, historical QoS data may not be the exact representative of the bids, but it is very similar to what service providers are willing to offer. In [18], Comuzzi et al. also utilise historical QoS data for evaluating service providers' QoS offers. The second threat to the external validity of our evaluation is the adoption of random distribution in the generation of bids. Bidders' bidding strategies can be very diverse. Thus, we adopted random distribution when generating bids in the experiments to provide a generalised reference. In scenarios where some service providers' bidding strategies follow other probability distributions rather than random distributions, or do not even follow any probability distributions, the results would be different from our evaluation in terms of actual figures. However, as the stringency of quality constraints vary, the change in effectiveness and efficiency of our approach in those scenarios would still be similar to our evaluation in general.

Threats to internal validity. The main threat to the internal validity of our evaluation is the comprehensiveness of our experiments. Three factors have been explored to simulate different stringency of quality constraints, including the number of quality constraints, the number of abstract services and the number of bidders. There is another factor that could influence the results - the maximum number of combinations that a bidder can bid for. In the experiments, it is equivalent to the number of abstract services. Changes in this factor will change the search space of the WDP, and as a result will influence the evaluation. However, the impact due to changes in this factor is the same as that due to changes in the number of bidders – they both increase the search space of the WDP. Given the fact that in the current experiments the maximum number of bids in the search space of the WDP is 60,000 which is quite large, we believe that we need not further change the maximum number of bids that a bidder can bid for in the experiments. In the experiments, we simulated scenarios where the abovementioned three factors changed individually. We also simulated scenarios where all three factors changed at the same time. Other scenarios that could have been simulated are those where the three factors change in pairs. In those scenarios, the results can be predicted in general based on the results that we have obtained. For example, if the numbers of quality constraints and bidders increase at the same time, the declining trend of the success rate would be similar to Fig. 9. The reason is that, based on Fig. 9 and Fig. 10, we can observe that the impact of the increase in the number of quality constraints on the success rate is much more significant than the number of bidders.

Threats to conclusion validity. The main threat to the conclusion validity of our evaluation is the statistical tests performed. We drew our conclusion from comparison by chi-square tests when evaluating the effectiveness of CASS. In such tests, large samples tend to result in a small *p*-value, lowering the practical significance of the test results [37]. There were 2,700 samples in each of the chi-square tests performed in our evaluation. This number is not even close to the numbers of samples that concern Lin et al. in [37]. The threat to the conclusion validity due to large samples in statistical tests, although potentially remains, is not significant. Thus, our evaluation has high conclusion validity.

8 Related Work

One major benefit of service-oriented paradigm is its support for building SBSs through composition of services. In the composition process, SBS designers need to select services that fulfil quality constraints and achieve SBS owners' optimisation goals for the SBSs. To address this issue, the research area of quality-aware service selection for SBSs has been attracting tremendous research attention in recent years.

The work in [8] proves that the service selection with multiple quality constraints for service composition is NP-complete. To address this issue, some suboptimal and optimal solutions have been proposed. We first summarise major existing approaches in the two categories. Then we introduce the common limitations of existing approaches that we overcome with CASS.

Suboptimal Approaches: The work in [12] uses genetic algorithms to address the issue of quality-aware Web service composition. Their work focuses on domain-specific QoS attributes and customised QoS aggregation formulas. WS-Binder Tool is implemented to support both cross-domain and domain-specific QoS attributes and to determine suboptimal solutions for Web service compositions according to given fitness functions and QoS constraint sets. However, the approach aims at providing service consumers with tools for domain-specific QoS definition and (re)binding, and no experimental results are provided for the evaluation of the approach. The work in [29] models quality-aware Web service selection as a 0-1 knapsack problem or resource constraint project scheduling problem and identifies four possible approaches to find near-optimal solutions, including greedy selection, discarding subsets, bottom-up approximation and pattern-wise selection. The work in [58] also models quality-aware Web service selection as a 0-1 knapsack problem as well as a multi-constraint optimal path problem. Yu et al. present heuristic algorithms to find near-optimal solutions in polynomial time. For different composition structures, e.g. sequence, parallel, conditional and loops of services, different algorithms are proposed. Berbner et al. [5] design an Web service-based workflow engine named WSQoSX, which aims at optimising quality-aware service composition in real-time and under heavy load. A heuristic algorithm is proposed. Firstly, linear programming is used to relax the Mixed Integer Programming (MIP) formulation of the service composition problem. Then a backtracking algorithm is used to construct a feasible solution based on the result of the relaxed IP problem. Liang et al. [36] propose to introduce a layer between QoS and service consumers' requirements in order to provide service consumers the QoS control of service systems. They adopt a negotiation approach to help select heuristically best services for optimising the overall system utility. Comuzzi and Pernici [19] propose a framework for automating Web service contract specification and establishment. Within the framework, functionally equivalent services are ranked according to their ability to fulfil the quality requirements within the target budget. Services are selected heuristically according to their ranks for target SBSs. A negotiation approach is also provided to maximise the overall system utility. Klein et al. [31] aim at optimising the quality of SBSs over long-term periods. At runtime, services are selected probabilistically for different executions of a same service composition schema. Aiming at finding near-optimal solutions in polynomial time, the authors relax the original SBS optimisation problem and then model the modified problem as IP problems. In [11, 13], with the aim to accommodate the aggregation of non-linear and domain-specific user-defined QoS parameters, Canfora et al. adopt genetic algorithms to find near-optimal solutions to the problem of quality-aware service selection for SBSs.

Optimal Approaches: Zeng et al. [59, 60] present AgFlow, a middleware platform that enables quality-driven composition of Web services. The selection of component service is performed to meet the users' requirements for the composite service's QoS modelled from multiple dimensions. IP is used to compute the optimal plan for composite service executions from several execution paths represented by Directed Acyclic Graph (DAG). Following the work in [59, 60], in [4], Ardagna and Pernici formulate the quality-aware service selection problem as MIP and adopt loops peeling for optimisation. When a feasible solution does not exist, a QoS negotiation algorithm is suggested to enlarge the solution space of the optimisation problem. Alrifai and Risse [2] adopt a heuristic distributed method to find the best Web services that meet local QoS constraints generated by decomposing global QoS constraints using, again, IP. They then propose in [3] an approach based on the notion of skyline to reduce the search space for the problem of quality-aware service composition. In [35], Li et al. use a different philosophy from works described above to address the quality-aware service selection problem. They use Service Composition Graph (SCG) to represent the composite service. Then, they employ Dijkstra's shortest-path algorithm to find the optimal solution to the service composition problem. In [52], Wang et al. find that optimal solutions can be found in polynomial time for some specially structured service compositions that consist of three services. Algorithms are proposed to detect if the optimal solution can be found for a given service composition in polynomial time. Attempting to maintain the optimality of SBSs at runtime, Cardellini et al. [14, 15] propose MOSES (Model-based Self-adaptation of SOA systems), a methodology and prototype that model the problem of service selection for SBS adaptation also as IP problems.

The abovementioned research does not fully consider the fact that complementary services can be provided at better QoS levels by a single provider than multiple providers. SBS designers can improve the system optimality of their SBSs by exploring the complementarity between the services. Exploring the complementarity between the services also enables attempts to increase the possibility of finding a solution for SBS optimisation problem, especially in scenarios where the quality constraints for SBSs are severe.

Exploiting the competition among service providers can also contribute to increasing the system optimality and the success rate of finding a solution for SBS optimisation problem. Thus, for an SBS, an SBS designer should be able to negotiate with multiple candidate service providers for each abstract service and select the best offers based on the results of the negotiation. In addition, the QoS that service providers are willing to offer depend on the complementarities between the services and the number of services that they are competing for. Therefore, in the negotiation, the service providers should be able to express their QoS offers flexibly for services and combinations of services. None of the existing negotiation approaches [4, 6, 19, 36] has properly addressed the above issues.

Combinatorial auction is widely recognised as a promising approach to provide the bidders (i.e. service providers) with flexible bidding options, hence allowing the bidders to compete in SLA negotiation for service composition [20]. The work in [41] presents a solution for QoS driven Web service composition using combinatorial auctions, named QoS-Aware WEb Services Composer (QWESC). QWESC facilitates combinatorial auctions to capture the Web service providers' willingness to provide a composite service at a lower price than the total price of stand-alone services that form the composite service, as well as other QoS attributes. IP formulation is provided to solve the problem of quality-aware Web service composition. However, several essential issues in designing a combinatorial auction for Web service composition are missing, e.g. bidding constraints and ask-QoS generation. The work presented in [46] also adopts combinatorial auction to model and solve the Web service composition problem. The proposed algorithms consider the volume discounts offered by service providers for multiple instances of Web services. Nonetheless, the approach has a major limitation that the issue of QoS constraints has not been considered. CWSMBM (Combinatorial Web Service Market Based Mechanism) utilises combinatorial auctions in a different way [38]. In CWSMBM, service consumers bid for services and the allocation is based on the utilities of the bids. Given the feature that it aims at improving the global market benefit, CWSMBM is only suitable for building applications within organisational boundaries, as claimed so by the authors themselves in the paper. In [6], Blau et al. design a multidimensional procurement auction for service compositions based on combinatorial auction. In the auction, a service provider can either offer a service on its own or provide bundled services together with other services. The complementarity between services is considered but roughly referred to as "synergy effect". An in-depth investigation into the complementarity between services, e.g., how to express and exploit the complementarity, is missing. In addition, the possibility that a service provider might be able to offer multiple services is not considered. Furthermore, since the focus of that research is the collaboration between service providers, the competition between them is not taken into account.

Combinatorial Auction for Service Selection (CASS) presented in this paper provides a novel approach that supports effective and efficient service selection for SBSs based on iterative multi-attribute combinatorial auction. CASS differs from the above works in two ways. First, CASS fully considers the complementarities between the services and allows service providers to express QoS offers flexibly for combinations of services. Second, CASS provides SBS designers the ability to exploit the competition among service providers.

9 Conclusions and Future Work

Quality-aware service selection for service-based systems (SBSs) is a critical issue in service-oriented environments. This paper has proposed Combinatorial Auction for Service Selection (CASS), a novel approach that supports effective and efficient service selection for SBSs based on iterative multi-attribute combinatorial auction. CASS properly captures the complementarities between services. CASS allows service providers to bid for combinations of services and express their quality-of-service (QoS) offers flexibly, giving them the incentives to compete. When a solution cannot be found, CASS iterates, allowing service providers to improve their bids. CASS also allows SBS designers to specify optimisation goals flexibly and models the problem of service selection for SBSs as constraint optimisation problem, which can be solved efficiently by adopting standard IP techniques. In CASS, auctioneers can create and exploit competition among service providers. The experimental evaluation shows high effectiveness of CASS. CASS beats existing IP based SBS optimisation approaches in success rate of finding a solution for SBS by an average of 42.96%. CASS also demonstrates high efficiency. On average, the overall system utility obtained by CASS is 1.41 times the overall system utility obtained by existing IP based optimisation approaches. The experimental results also show that the auction duration and coordination overhead of CASS are satisfactory.

One of the limitations of CASS is its inability to handle non-linear quality constraints and non-linear QoS parameters. With non-linear quality constraints and QoS parameters, the WDP becomes a non-linear programing problem. We will investigate this in the future. We will also further consider the impact of different bidder behaviours on CASS in terms of effectiveness and efficiency.

Acknowledgment

This work is partly funded by the Australian Research Council under grant No. DP0663841 and LP0990393, National Science Foundation of China under grant No. 90412010 and China Grid project from Ministry of Education of China. This paper is a significant revision and extension of “INSC: An Iterative Negotiation Approach for Service Composition”, published in the proceedings of the 2012 IEEE International Conference on Services Computing (SCC2012), 24-29 June 2012, Honolulu, Hawaii, USA. Y. Yang is the corresponding author for the paper.

REFERENCES

[1] E. Al-Masri and Q. H. Mahmoud, "Investigating Web Services on the World Wide Web," *Proc of 17th International Conference on World Wide Web (WWW2008)*, pp. 795-804, 2008.

[2] M. Alrifai and T. Risse, "Combining Global Optimization with Local Selection for Efficient QoS-Aware Service Composition," *Proc of 18th International Conference on World Wide Web (WWW2009)*, Madrid, Spain, pp. 881-890, 2009.

[3] M. Alrifai, D. Skoutas, and T. Risse, "Selecting Skyline Services for QoS-based Web Service Composition," *Proc of 19th International Conference on World Wide Web (WWW2010)*, Raleigh, North Carolina, USA, pp. 11-20, 2010.

[4] D. Ardagna and B. Pernici, "Adaptive Service Composition in Flexible Processes," *IEEE Transactions on Software Engineering,* vol. 33, no. 6, pp. 369-384, 2007.

[5] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, "Heuristics for QoS-aware Web Service Composition," *Proc of IEEE International Conference on Web Services (ICWS2006)*, Chicago, Illinois, USA, pp. 72-82, 2006.

[6] B. Blau, T. Conte, and C. van Dinther, "A Multidimensional Procurement Auction for Trading Composite Services," *Electronic Commerce Research and Applications,* vol. 9, no. 5, pp. 460-472, 2010.

[7] P. Bonami, M. Kilinç, and J. Linderoth, "Algorithms and Software for Convex Mixed Integer Nonlinear Programs," *Mixed Integer Nonlinear Programming,* vol. 154, pp. 1-39, 2012.

[8] P. A. Bonatti and P. Festa, "On Optimal Service Selection," *Proc of 14th International Conference on World Wide Web (WWW2005),* Chiba, Japan, pp. 530-538, 2005.

[9] C. Boutilier and H. H. Hoos, "Bidding Languages for Combinatorial Auctions," *Proc of 17th International Joint Conference on Artificial Intelligence (IJCAI2001)*, Seattle, Washington, USA, pp. 1211-1216, 2001.

[10] P. Brewer and C. Plott, "A Binary Conflict Aascending Price (BICAP) Mechanism for the Decentralized Allocation of the Right to Use Railroad Tracks," *International Journal of Industrial Organization,* vol. 14, no. 6, pp. 857-886, 1996.

[11] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "A Framework for QoS-Aware Binding and Re-Binding of Composite Web Services," *Journal of Systems and Software,* vol. 81, no. 10, pp. 1754-1769, 2008.

[12] G. Canfora, M. D. Penta, R. Esposito, F. Perfetto, and M. L. Villani, "Service Composition (re)Binding Driven by Application-Specific QoS," *Proc of 4th International Conference on Service-Oriented Computing (ICSOC2006)*, Chicago, IL, USA, pp. 141-152, 2006.

[13] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "QoS-Aware Replanning of Composite Web Services," *Proc of IEEE International Conference on Web Services (ICWS2005)*, Orlando, FL, USA, pp. 121-129, 2005.

[14] V. Cardellini, E. Casalicchio, V. Grassi, S. Lannucci, F. Lo Presti, and R. Mirandola, "MOSES: A Framework for QoS Driven Runtime Adaptation of Service-Oriented Systems," *IEEE Transactions on Software Engineering,* vol. 38, no. 5, pp. 1138-1159, 2012.

[15] V. Cardellini, E. Casalicchio, V. Grassi, F. Lo Presti, and R. Mirandola, "QoS-driven Runtime Adaptation of Service Oriented Architectures," *Proc of 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (SEC/SIGSOFT FSE2009)*, Amsterdam, The Netherlands, pp. 131-140, 2009.

[16] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. *Web Services Description Language (WSDL) 1.1*. URL: http://www.w3.org/TR/wsdl, 2001.

[17] L. Clement, A. Hately, C. von Riegen, and T. Rogers. *UDDI Version 3.0.2*. URL: http://www.uddi.org/pubs/uddi_v3.htm, 2004.

[18] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, and R. Yahyapour, "Establishing and Monitoring SLAs in complex Service Based Systems," *Proc of IEEE International Conference on Web Services (ICWS2009)*, Los Angeles, CA, USA, pp. 783-790, 2009.

[19] M. Comuzzi and B. Pernici, "A Framework for QoS-Based Web Service Contracting," *ACM Transactions on The Web,* vol. 3, no. 3, pp. 1-52, 2009.

[20] P. S. Cramton, Y. and R. Steinberg, *Combinatorial Auctions*. Cambridge, Mass.: MIT Press, 2006.

[21] C. D'Ambrosio and A. Lodi, "Mixed Integer Nonlinear Programming Tools: A Practical Overview," *4OR,* vol. 9, no. 4, pp. 329-349, 2011.

[22] E. Di Nitto, M. Di Penta, A. Gambi, G. Ripa, and M. L. Villani, "Negotiation of Service Level Agreements: An Architecture and a Search-Based Approach " *Proc of 5th International Conference on Service-Oriented Computing (ICSOC2007)*, Vienna, Austria, pp. 295-306, 2007.

[23] S. Dulluri and N. R. S. Raghavan, "Allocation of Advertising Space by a Web Service Provider Using Combinatorial Auctions," *SADHANA,* vol. 30, no. 2, pp. 213-230 2005.

[24] D. Durkee, "Why Cloud Computing Will Never Be Free," *Communications of the ACM,* vol. 53, no. 5, pp. 62-69, 2010.

[25] M. Gillmann, G. Weikum, and W. Wonner, "Workflow Management with Service Quality Guarantees," *Proc of ACM SIGMOD International Conference on Management of Data*, Madison, Wisconsin, USA, pp. 228-239, 2002.

[26] L. M. V. Gonzalez, L. Rodero-Merino, C. Juan, and M. A. Lindner, "A Break in the Clouds: Towards A Cloud Definition," *Computer Communication Review,* vol. 39, no. 1, pp. 50-55, 2009.

[27] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar, and Y. Lafon. *SOAP Version 1.2*. URL: http://www.w3.org/TR/soap12-part1/, 2007.

[28] J. Hoffmann, P. Bertoli, and M. Pistore, "Web Service Composition as Planning, Revisited: In Between Background Theories and Initial State Uncertainty," *Proc of 22nd AAAI Conference on Artificial Intelligence (AAAI2005)*, Vancouver, British Columbia, Canada, pp. 1013-1018, 2007.

[29] M. C. Jaeger, G. Mühl, and S. Golze, "QoS-Aware Composition

of Web Services: A Look at Selection Algorithms," *Proc of IEEE International Conference on Web Services (ICWS2005)*, Orlando, FL, USA, pp. 807-808, 2005.

[30] R. Khalaf, N. Mukhi, and S. Weerawarana, "Service-Oriented Composition in BPEL4WS," *Proc of 12th International Conference on World Wide Web (WWW2003)*, Budapest, Hungary, 2003.

[31] A. Klein, F. Ishikawa, and S. Honiden, "Efficient QoS-Aware Service Composition with a Probabilistic Service Selection Policy," *Proc of 8th International Conference on Service-Oriented Computing (ICSOC2010)*, San Francisco, CA, USA, pp. 182-196, 2010.

[32] S. Kona, A. Bansal, and G. Gupta, "Automatic Composition of Semantic Web Services," *Proc of International Conference on Web Services (ICWS2007)*, Salt Lake City, Utah, USA, pp. 150-158, 2007.

[33] U. Küster, B. König-Ries, M. Stern, and M. Klein, "DIANE: An Integrated Approach to Automated Service Discovery, Matchmaking and Composition," *Proc of 16th International Conference on World Wide Web (WWW2007)*, Banff, Alberta, Canada, pp. 1033-1042, 2007.

[34] J. Ledyard and K. Szakaly, "Designing Organizations for Trading Pollution Rights," *Journal of Economic Behavior and Organization*, vol. 25, no. 2, pp. 167-196, 1994.

[35] Y. Li, J. Huai, T. Deng, H. Sun, H. Guo, and Z. Du, "QoS-aware Service Composition in Service Overlay Networks," *Proc of IEEE International Conference on Web Services (ICWS2007)*, Salt Lake City, Utah, USA, pp. 703-710, 2007.

[36] Q. Liang, X. Wu, and H. C. Lau, "Optimizing Service Systems Based on Application-Level QoS " *IEEE Transactions on Services Computing*, vol. 2, no. 2, pp. 108-121, 2009.

[37] M. Lin, H. C. Lucas Jr., and G. Shmueli, "Too Big to Fail: Large Samples and the p-Value Problem," *Information Systems Research*, 2013.

[38] S.-Y. Lin, B.-Y. Chen, C.-C. Liu, and V.-W. Soo, "Web Service Allocations based on Combinatorial Auctions and Market-based Mechanisms," *Proc of 12th International Conference on CSCW in Design (CSCWD2008)*, Xi'an, China, pp. 452-458, 2008.

[39] D. A. Menascé, "QoS Issues in Web Services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 72-75, 2002.

[40] P. Milgrom, "The Structure of Information in Competitive Bidding," PhD Thesis, School of Humanities and Sciences, Stanford University, 1979.

[41] M. Mohabey, Y. Narahari, S. Mallick, P. Suresh, and S. V. Subrahmanya, "A Combinatorial Procurement Auction for QoS-Aware Web Services Composition," *Proc of 3rd IEEE Conference on Automation Science and Engineering (CASE2007)*, Scottsdale, AZ, USA, pp. 716-721, 2007.

[42] S. Mumtaz, A. Villazon, and T. Fahringer, "Grid Allocation and Reservation - Grid Capacity Planning with Negotiation-based Advance Reservation for Optimized QoS," *Proc of ACM/IEEE Conference on High Performance Networking and Computing (SC2006)*, Tampa, FL, USA, p. 103, 2006.

[43] OASIS. *Web Services Business Process Execution Language Version 2.0.* URL: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf, 2007.

[44] Object Management Group. *Business Process Model And Notation (BPMN) Version 2.0.* URL: http://www.omg.org/spec/BPMN/2.0/PDF/, 2011.

[45] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso, "Automated Composition of Web Services by Planning at the Knowledge Level," *Proc of 19th International Joint Conference on Artificial Intelligence (IJCAI2005)*, Edinburgh, Scotland, UK, pp. 1252-1259, 2005.

[46] B. Prashanth and Y. Narahari, "Efficient Algorithms for Combinatorial Auctions with Volume Discounts Arising in Web Service Composition," *Proc of 4th IEEE Conference on Automation Science and Engineering (CASE2008)*, Arlington, VA, pp. 995-1000, 2008.

[47] D. Quan, "Real Estate Auctions: A Survey of Theory and Practice," *Journal of Real Estate Finance and Economics*, vol. 9, no. 1, pp. 23-49, 1994.

[48] A. Ragone, T. D. Noia, E. D. Sciascio, F. M. Donini, S. Colucci, and F. Colasuonno, "Fully Automated Web Services Discovery and Composition through Concept Covering and Concept Abduction," *International Journal of Web Services Research*, vol. 4, no. 3, pp. 85-112, 2007.

[49] S. Rassenti, V. Smith, and B. R., "A Combinatorial Auction Mechanism for Airport Time Slot Allocation," *Bell Journal of Economics*, vol. 13, no. 2, pp. 402-417, 1982.

[50] T. Sandholm, "Algorithm for Optimal Winner Determination in Combinatorial Auctions," *Artificial Intelligence*, vol. 135, no. 1-2, pp. 1-54, 2002.

[51] T. Sandholm, S. Suri, A. Gilpin, and D. Levine, "Winner Determination in Combinatorial Auction Generalizations," *Proc of The 1st International Joint Conference on Autonomous Agents & Multiagent Systems (AAAMS2002)*, Bologna, Italy, pp. 69-76, 2002.

[52] J. Wang, J. Wang, B. Chen, and N. Gu, "Minimum Cost Service Composition in Service Overlay Networks," *World Wide Web*, vol. 14, no. 1, pp. 75-103, 2011.

[53] W.-L. Wang, D. Pan, and M.-H. Chen, "Architecture-based Software Reliability Modeling," *Journal of Systems and Software*, vol. 79, no. 1, pp. 132-146, 2006.

[54] M. Wellman, W. Walsh, P. Wurman, and J. MacKie-Mason, "Some Economics of Market-Based Distributed Scheduling," *Proc of 18th International Conference on Distributed Computing Systems (ICDCS1998)*, Amsterdam, The Netherlands, pp. 612-621, 1998.

[55] R. Wilson, "A Bidding Model of Perfect Competition," *Review of Economic Studies*, vol. 44, no. 3, pp. 511-518, 1977.

[56] J. Yan, R. Kowalczyk, J. Lin, C. M. B., S. Goh, and J. Y. Zhang, "Autonomous Service Level Agreement Negotiation for Service Composition Provision," *Future Generation Computer Systems*, vol. 23, no. 6, pp. 748-759, 2007.

[57] K. Yoon and C.-L. Hwang, *Multiple Attribute Decision Making: An Introduction (Quantitative Applications in the Social Sciences)*: SAGE, 1995.

[58] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints," *ACM Transactions on the Web*, vol. 1, no. 1, 2007.

[59] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality Driven Web Services Composition," *Proc of 12th International Conference on World Wide Web (WWW2003)*, Budapest, Hungary, pp. 411-421, 2003.

[60] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services

Composition," *IEEE Transactions on Software Engineering,* vol. 30, no. 5, pp. 311-327, 2004.

[61] Z. Zheng and M. R. Lyu, "Collaborative Reliability Prediction of Service-Oriented Systems," *Proc of 32nd ACM/IEEE International Conference on Software Engineering (ICSE2010)*, Cape Town, South Africa, pp. 35-44, 2010.

APPENDIX

Acronym Summary

See Table 6

TABLE 6
ACRONYM SUMMARY

Symbol	Description
CASS	**C**ombinatorial **A**uction for **S**ervice **S**election
CSP	**C**onstraint **S**atisfaction **P**roblem
COP	**C**onstraint **O**ptimisation **P**roblem
DMD	**D**ynamic **M**inimum **D**ecrement
IP	**I**nteger **P**rogramming
MIP	**M**ixed **I**nteger **P**rogramming
RFP	**R**equest **f**or **P**roposal
SBS	**S**ervice-**b**ased **S**ystem
SLA	**S**ervice **L**evel **A**greement
QoS	**Q**uality **o**f **S**ervice
UDDI	**U**niversal **D**escription **D**iscovery and **I**ntegration
WDP	**W**inner **D**etermination **P**roblem

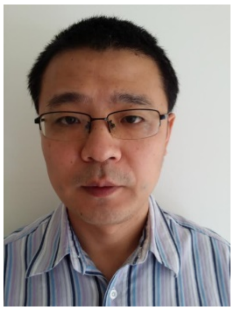
Notion Summary

See Table 7

TABLE 7
NOTATION SUMMARY

Symbol	Description
AQ	Ask-QoS
$aq_{i,p}(s_j)$	Asked offer for the pth QoS parameter of s_j specified in the ith ask-QoS
$\mathscr{B}$	Universe of current bids
b_i	ith bid proposed by a bidder
br_k	kth bidder
BR	Universe of bidders
cb	Conditional branch
c_p	pth constraint
C	Set of constraints
$D(q(s))$	Domain of the qth QoS parameter of service s
$ep(es_e)$	Execution probability of execution scenario es_e
e	Execution scenario index
es_e	eth execution scenario
EP	Eexecution path
k	Bidder index
K	Universe of items to be auctioned
s	Service
SC	Service Class
S	Service-based system
S_i	Set of services
Q	Set of QoS parameters
Q_i	Set of QoS parameters for S_i
p	Quality parameter index
$prob_i$	Probability that a loop iterates for i times
$prob(cb_i)$	Probability that cb_i is selected for execution
q_p	pth QoS parameter
$q_p^{max}(SC_i)$	Maximum value for the pth QoS parameter in the ith service class
$q_p^{min}(SC_i)$	Minimum value for the pth QoS parameter in the ith service class
$q_p^{k,cur}(s_i)$	pth QoS parameter of s_i proposed by bidder br^k in the current round
$q_p^{k,1}(s_i)$, …, $q_p^{k,n}(s_i)$	Series of br^k's historical offers for $q_p(s_i)$
$q_p^{k,ave}(s_i)$	Average value of $q_p^{k,1}(s_i)$, …, $q_p^{k,n}(s_i)$
u	Utility function
w	Weight
X	Set of variables
X_i	ith variable in X
z	Standard score
$\varepsilon_p^k(s_i)$	Surplus bidding space of br_k for $q_p(s_i)$

He received the B.Eng. degree in computer science and technology from Huazhong University of Science and Technology, his first Ph. D. degree in information and communication technology from Swinburne University of Technology (SUT), Australia, in 2009 and his second Ph. D. degree in computer science and engineering from Huazhong University of Science and Technology (HUST), China, in 2010. He is now a research fellow at SUT. His research interests include services computing, cloud computing, P2P system, workflow management and agent technologies.



Jun Han received the B.Eng. and M.Eng. degrees in computer application technologies from Southeast University, Nanjing, China, in 1998 and 2001, respectively, and the Ph.D. degree in information technology from Swinburne University of Technology, Melbourne, Vic., Australia, in 2004. He is currently a Senior Lecturer in School of Information Systems and Technology, University of Wollongong, Wollongong, NSW, Australia. His research interests include process management and workflow technology, Web services, Cloud computing, Internet computing, and e-commerce.

Hai Jin is a Cheung Kung Scholars Chair Professor of computer science and engineering at the Huazhong University of Science and Technology (HUST) in China. He is now Dean of the School of Computer Science and Technology at HUST. Jin received his PhD in computer engineering from HUST in 1994. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. Jin worked at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. Jin is the chief scientist of ChinaGrid, the largest grid computing project in China, and the chief scientist of National 973 Basic Research Program Project of Virtualization Technology of Computing System. Jin is a senior member of the IEEE and a member of the ACM. Jin is the member of Grid Forum Steering Group (GFSG). He has co-authored 15 books and published over 400 research papers. His research interests include computer architecture, virtualization technology, cluster computing and grid computing, peer-to-peer computing, network storage, and network security Jin is the steering committee chair of International Conference on Grid and Pervasive Computing (GPC), Asia-Pacific Services Computing Conference (APSCC), International Conference on Frontier of Computer Science and Technology (FCST), and Annual ChinaGrid Conference. Jin is a member of the steering committee of the IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), the IFIP International Conference on Network and Parallel Computing (NPC), and the International Conference on Grid and Cooperative Computing (GCC), International Conference on Autonomic and Trusted Computing (ATC), International Conference on Ubiquitous Intelligence and Computing (UIC).

Yun Yang was born in Shanghai, China. He received the B.Sci. degree from Anhui University, Hefei, China, in 1984, the M.Eng. degree from the University of Science and Technology of China, Hefei, China, in 1987, and the Ph.D. degree from the University of Queensland, Brisbane, Australia, in 1992, all in computer science. He is currently a Full Professor in the Faculty of Information and Communication Technologies at Swinburne University of Technology, Melbourne, Australia. Prior to joining Swinburne as an Associate Professor, he was a Lecturer and Senior Lecturer at Deakin University during 1996-1999. Before that, he was a (Senior) Research Scientist at DSTC Cooperative Research Centre for Distributed Systems Technology during 1993-1996. He also worked at Beihang University during 1987-1988. He has co-authored four monographs and published more than 200 papers on journals and refereed conferences. His current research interests include software technologies, cloud computing, workflow systems, big data, and service-oriented computing.