An Agent Service Brokering Algorithm for Winner Determination in Combinatorial Auctions

Aneurin M. Easwaran[1] and Jeremy Pitt[1]

Abstract. Deregulation of telecommunications has meant an increase in third-party service provision, personalized service delivery and integrated networks and media. The efficient allocation of services, without human intervention, to satisfy advanced service requirements spanning several networks is a crucial task. This can be modeled as a winner determination problem in combinatorial auctions where there are multiple services, service providers and winner determination criteria (like cost, bandwidth, delay, etc) but we have shown the problem is NP-complete.

This paper describes a new two-stage algorithm for optimal anytime winner determination. In the first stage, a hierarchical task network planner is used to decompose a task into subtasks that can be solved by the available services. In the second stage, a genetic algorithm with heuristics is used to find the optimal combination of service providers to provide the services identified. We present our algorithm used to solve the second stage in detail and the results from various experiments. The results show the GA finds optimal solutions much quicker than a modified depth-first search with pruning. We also show the genetic algorithm a) finds optimal solutions quicker when deal lengths have a random distribution and b) initial anytime performance is better when deal lengths have an exponential distribution.

1 INTRODUCTION

One of the basic problems of open, multi-agent systems for the Internet is the connection problem [9]. That is, each agent must be able to locate the other agents who may have capabilities which are necessary for the execution of tasks. The solution to this problem relies on using some 'well-known' agents and some basic interactions with them – matchmaking or brokering.

We have developed a brokering system where there are client agents, service agents and one or more brokers. A client agent has a task which can be solved by a single service agent or a federation/combination of service agents. However, the client is not aware of what service agents are available or how a task can be, necessarily, decomposed and solved by a set of service agents in the best possible way. The client agent requests the agent service broker to "recommend" a set of service agents who are capable of solving a particular task. The broker maintains a repository containing current and correct information about operational service agents. Each service agent represents a particular service provider. There can be one or more service agents (providers) providing the same service. The broker identifies a combination of service agents who have the capabilities to solve a client's task. The broker decomposes a task into subtasks and one or more service agents have the capability to solve each subtask. The service agents who provide the same service may vary in various measures like cost, bandwidth, delay, etc. The service agents who provide different services may collaborate in order to enhance some measure. The broker selects winners to solve a task based on multiple criteria/measure and ensures the overall measure of the set of winners is optimal or near-optimal. The client agent is notified by the broker of the service agents that are required to solve the task. The client is then free to initiate a dialogue with the service agents for the appropriate services. This paper focuses on brokering a combination of services to solve a task.

Combinatorial auctions are auctions where bidders can bid on combinations of items. Combinatorial auctions are applicable to many real world situations. In an auction for the right to use railroad segments a bidder desires a bundle of segments that connect two particular points; at the same time, there may be alternate paths between these points and the bidder needs only one[1]. A set of services can be combined to improve the cost or quality of service of the combination. The services can be combined in many ways. For example, there can be a deal where one can buy a service and gets another service for free and, therefore, the combination of services would have a cost equal to the cost of one service. Obviously it is cheaper to buy the combination than the individual services. The broker considers the various combinations in order to find the optimal solution (i.e. a set of service providers) based on multiple criteria. In combinatorial auctions bidders may place bids on combination of items whereas in our system service providers quote a particular value for a combination of services.

While economics and game theory provide many insights into the potential use of such auctions, they have little to say about computational considerations. In this paper we address the computational complexity of combinatorial auctions.

2 WINNER DETERMINATION

In essence, the winner determination problem is to find an optimal set of service agents to solve a client's task - optimal in terms of criteria specified. The problem has two parts – satisfiability and optimization.

Satisfiability: Given a set of services and a task, establish

[1] Intelligent and Interactive Systems, Department of Electrical & Electronic Engineering,
Imperial College of Science, Technology & Medicine,
Exhibition Road, London SW7 2BZ, England.
E-mail: a.easwaran@ic.ac.uk

whether the task is satisfiable by the current set of services available. Basically identifying the services required to solve a task. This involves breaking a task into subtasks until there are only primitive tasks that can be solved by a combination of service agents.

Optimization: Identify a set of winners or service providers based on multiple criteria, to provide the services identified to solve the client's task. This paper mainly focuses on the optimization stage of winner determination.

The broker must consider the various relationships that exist between services (where the respective service providers can be the same or different) in order to identify a set of service agents to solve a task at the optimum measure(s). The service providers identified by the broker to solve a task are the winners. The relationship between services is one of three types:

- *Cooperation:* Different services can work together regardless of whom provides each service.
- *Benefit cooperation:* Different services can work together regardless of whom provides each service but there is an added benefit for using particular service providers for the required services. Service combinations or deals are the result of this relationship.
- *No cooperation:* Different services cannot work together regardless of whom provides each service.

The *order* in which services are combined matters in all three relationships. For example, it may not be possible to combine (no cooperation relationship) serviceA with serviceB but combining serviceB with serviceA may have an added benefit (either a cooperation or a benefit cooperation relationship). Ordering of services is particularly important in the telecommunications domain. These relationships are quantified according to the criteria used for optimizing. For example, if winner determination is based on cost of services then a no cooperation relationship can be represented by a very large number (or ∞). The relationship between services grows exponentially as the number of services and service providers increase.

Proposition 2.1: Winner determination is NP-complete.

Proof: The Travelling Salesman model assumes that the decision-maker has determined a priori which cities will be sequenced. Consider a generalization of the TSP, which combines the decisions of city selection and city sequencing. Instead of pre-selecting the cities to be visited, the generalized model assumes the cities have been grouped into mutually exclusive and exhaustive states. The generalized travelling salesmen problem (GTSP)[10] is then to find a minimum cost path which includes exactly one city from each state. The GTSP is transformed to an agent service brokering problem. Each state is transformed to a service and each city in a state is transformed to a service provider providing the service. The cost of travelling between two states is transformed to cost of purchasing two services. The generalized travelling salesmen problem is NP-complete. A detail proof can be found in [4]. ■

The search space for winner determination is greater than the GTSP when we take into account the number of possible ways in which a set of services can be combined in a route. In the GTSP, adding a new node to a route increases the total cost of the route by the amount required to travel from the last node (n-1) to the new node (n). Whereas in the agent service brokering problem, adding a new node does not necessarily increase the total cost of the route by the amount required to purchase the new node because the cost of adding a new node is dependent on the history of the route. For example, say, there is a route with four services ($S_1S_2S_3S_4$) and the total cost of the route is the sum of two deals (i.e. benefit cooperation relationship – S_1S_2 and S_3S_4). Now adding a new service (S_5) to the route may bring into effect two different deals ($S_1S_2S_3$ and S_4S_5) which may be cheaper than buying S_5 and the previous two deals - S_1S_2 and S_3S_4.

Proposition 2.2: The number of possible solutions to a problem is $(n! * (m_1 * \ldots * m_n) * 2^{(n-1)})$ where n is the number of services and m_x is number of service providers providing service x and there are $2^{(n-1)}$ possible service combinations. The number of possible solutions to a GTSP is $(n! * (m_1 * \ldots * m_n))$ where n is the number of states and m_x is number of cities in state x.

Proof: We outline a proof to show how $2^{(n-1)}$ is derived. Two services can be combined (AND operator) or not combined (OR operator). If there are n services then there are (n-1) places where either of the operators could appear. Since there are 2 operators and n-1 places where the operators can appear then there are $2^{(n-1)}$ ways in which n services can be combined. ■

3 ARCHITECTURE OVERVIEW

The broker consists of two components to solve each stage of the winner determination problem – *planner* and *optimizer*. The planner receives a task from a client which is decomposed into subtasks using predefined plan methods. The planner also identifies the services that are required to solve each subtask. The planner then notifies the optimizer of the services that are required to solve each subtask.

The planning process developed is based on a hierarchical task network planning formalism. The planner searches through plan space to solve the planning stage of the problem. Conventional wisdom in the planning community, supported to large extent by the fielded applications to-date, holds that most real world domains are best modeled with hierarchical task network planning models[7]. The planner starts with a task or goal, and on each iteration adds one more step i.e. decomposes a task into sub-tasks until there are only primitive tasks. It does this by choosing some operator – either from existing steps of the plan or from the pool of operators – that achieves a complex task. If this leads to an inconsistent plan, it backtracks and tries another branch of the search space. To keep the search focussed, the planner only considers adding steps that serve to achieve a complex task that has not yet been achieved. The operators are mainly various task decomposition methods which capture human expertise. We do not present any detail information on the planner due to space constraints.

The optimizer identifies the best combination of service providers to provide the services identified. In order to achieve this it considers the various service combinations (deals) with same or different service providers. The optimizer is a genetic algorithm (GA) with heuristics that yields an anytime algorithm.

4 GENETIC ALGORITHM AND HEURISTICS

The second stage of the brokering process involves selecting a combination of service providers to provide the identified services. This stage is solved to optimality or near optimality, depending on the size of the problem and the time available to the user, by a genetic algorithm[6]. The genetic algorithm is a highly

parallel mathematical algorithm that transforms a population of individual objects, each with an associated value of fitness, into a new generation of the population, using the Darwinian principle of survival and reproduction of the fittest and analogs of naturally occurring genetic operations such as crossover and mutation. Recently genetic algorithms[2] were applied to the knife change minimization problem, which is an instance of the GTSP, to produce good solutions.

Problem representation is critical to the success of GA. Each possible point in the search space of the problem must be encoded as a character string (i.e., as a chromosome). There are number of ways to represent a problem. The representation used by the GA is diagrammed in Figure 1. We represent each individual in the population using three chromosomes:

1. *Service chromosome* – Represents the services that are required to solve a task. Service combinations are based on the order in which the services appear in the chromosome.
2. *Provider chromosome* – Represents the service provider providing the service shown by the service chromosome. For example, service S_2 is provided by provider P_A.
3. *Deal chromosome* – Represents the services that can be combined to benefit in some way, e.g. cost. A service combination is represented using 1 or 0 and a non-combination is represented using *. For example, services S_2 and S_4 are combined to form a deal whereas service S_3 is not combined with any other service.

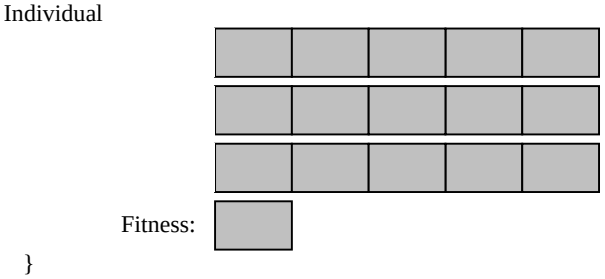

Figure 1: Problem Representation

Each individual has a fitness value associated to it. The fitness of an individual is the cost of purchasing the services but it can be the sum of one or more measures that are to be optimized.

The outline of the genetic algorithm is given in Figure 2. The GA initially creates a population using the data about the services required, their providers and the existing deals/combinations. Heuristics are applied on the initial population to improve the quality of the population. Individuals are selected from the old population to create a new population by applying the genetic operators on the selected individuals. The search process continues for a fixed number of generations or until there is no improvement in the quality of the best individual. Repair heuristics are applied on the best individual in an attempt to improve the quality of the solution further. The GA stores the best individual from the start of the search process and individual is replaced when a better one is found as the search progresses. The search process can be terminated at anytime and the current best individual would be the solution. Thus, the GA yields an anytime algorithm. Inevitably, optimizing on more than one criteria involves a trade-off between profit optimization and end-user satisfaction. The best solution found by the GA may not suit the client. To overcome this problem, the client can specify the number of solutions to be produced by the GA and then select the best. The time required by the GA to find 1 or *n* solutions is the same unlike other search methods.

Genetic operators are applied on the individuals to improve the fitness of the individuals from one generation to another. Simple crossover and mutation operators produce illegal chromosomes when applied. For example, the one-point crossover which replaces a certain proportion of a chromosome with an equal proportion of another chromosome may produce chromosomes where the services are repeated or the service provider for a service is wrong. To avoid such problems special crossover and mutation operators are to be applied on the chromosomes of the individuals selected. The selection of individuals in the GA are either random or fitness based (tournament). Mutation operators were developed to introduce new service ordering, new providers or new deals. The crossover operator in the GA is responsible for transferring a deal from one parent to another to create two new children. The crossover operator ensures that the deal that is transferred from one parent to another will have minimum impact on the existing deals in the other parent.

1. Get data on - Services, Providers and Deals.
2. Create initial population.
3. Apply initial population heuristics.
4. Select individuals: Random or fitness based.
5. On selected individuals apply genetic operators (mutation and crossover).
6. Add new individuals to new population. Calculate fitness of each individual.
7. Next Generation - Go to step 4.
8. Apply repair heuristics to final solution.

Figure 2: Genetic Algorithm Outline

Two types of heuristics were developed to improve the performance and the quality of the solution produced by the GA. The initial population heuristic was developed to capitalize heavily on the sparseness of deals. In practice the space of deals is necessarily extremely sparsely populated. For example, if there are 100 services, there are $2^{(100-1)}$ combinations, and it would take a very, very long time to create all those combinations. Therefore, randomly generating deals when creating the initial population is futile as most randomly generated deals would not match the existing valid deals (in the repository).

1. *Initial population heuristic* – The heuristic randomly selects valid deals/combinations from the repository and uses them to create new individuals instead of randomly creating combinations.
2. *Repair heuristic* – This heuristic ensures all services that are not part of a deal/combination have the best (cheapest) service provider.

The output of the GA is a list of service agents who are capable of solving the task. The user's previous experience with the service agents may classify one or more of the identified agents as faulty or not trustworthy. The user can then instruct the broker to find an *alternative* solution to the task by excluding a particular set of service agents. As expected, the new solution would be near-optimal or, possibly, optimal. The GA finds an alternative solution by solving the problem again without the specified set of service agents and with a selection of valid chromosomes from the final population of the previous run. Thus, the selection of chromosomes from the final population creates a good initial population for the GA.

5 MODIFIED DEPTH-FIRST SEARCH

The depth-first search algorithm examines all feasible deal and/or service combinations to find the optimal solution. The outline of the algorithm is given in Figure 3.

1. Get data on - Services, Providers and Deals.
2. Set length L = Number of services required (includes service repetition.)
3. Find a deal of length l, where $l = L$ initially.
4. For remaining search space equal to $(L\text{-}l)$ generate all deal and service combinations. Unnecessary combinations are pruned.
5. Find the cheapest set of deals and/or services to satisfy the various combinations generated.
6. Set $l = l$ (check for more deals of the same length) or $l = (l - 1)$, then go to step 3. Or terminate if $l = 0$.

Figure 3: Depth-First Search Outline

6 BROKERING SCENARIO AND RELATED WORK

The broker algorithm is applied to an abstract problem called the blocks world. The blocks world problem is adapted in order to demonstrate the broker functionality where each block represents a particular service. Associated with each block is a price, a colour and a level of softness. Each block has a connector and the connectors are of different shapes. Blocks can be combined if they have the same connector shape. The blocks are provided by one or more block providers and the block providers represent service providers. The type of task that is required to be solved in this domain is building a tower at the cheapest cost and with the lowest level of softness. The client specifies the size of the tower (i.e. number of blocks), the colours and the connector shape.

Several commercial and academic auction houses have recently appeared on the Internet, but to our knowledge, this implementation is the first of its kind – where there are multiple service providers, repetition of services in a deal, multiple optimization criteria and the bids are not superadditive: $cost(S_1 \cup S_2) \geq cost(S_1)+cost(S_2)$. Sandholm[8] and Fujishima et al.[5] have addressed the conventional combinatorial auction problem but their approach to solving the problem very different to our approach. Sandholm presented a Bidtree algorithm that performs a secondary depth-first search to identify non-conflicting bids. The Bidtree then uses an IDA* search strategy to solve the problem. Fujishima et al. have presented two methods for winner determination. The first method is a modified depth-first search applied on a structured search space to reduce runtime. Caching and pruning are also used to speed searching. The second method is a heuristic, market-based approach. It sets up a virtual multi-round auction in which a virtual agent represents each original bid bundle and places bids, according to a fixed strategy, for each good in the bundle.

7 EXPERIMENTAL SETUP AND RESULTS

We conducted various empirical tests to evaluate a) the general performance of the GA and the depth-first search algorithm, b) how the run time of the GA varies with different deal length/size distribution, and c) how percentage optimality of the GA varies with time, given a particular deal length distribution and a fixed number of deals and services. We executed our programs written in Java on a PC (Pentium 200MHz with 64 RAM) to get the results. All the results reported are averages over 25 different runs and the optimization was based on a single criteria i.e. cost. In the case of the GA, each run was terminated when the optimal solution was found. The run times include the planning stage of the brokering algorithm and the optimization is based on a single criteria i.e. minimization of cost of services. We do not present any experiments varying the number of services, service providers and optimization criteria in this paper due to space constraints.

The GA without any heuristics finds optimal solutions for small problems (10 services) but the heuristics are required if the GA is to find optimal solutions for large problems (100 services)[3]. In order to analyze the performance of the GA to different deal length distributions we had to generate valid deals to fit the distributions. In the absence of real data we tested our algorithms against deals randomly generated using the service data in the broker's repository. Each deal was created by randomly selecting services from 1..m services. Each service has 1 to 5 providers and an appropriate provider was also randomly selected to provide the selected service. The price of deals for n services is randomly distributed between $[c(1\text{-}d), c(1+d)]$ where c= sum($price_1$..$price_n$) and d=0.1. The size of deals (i.e. number of services in a deal) plays an important role in the complexity of the winner determination problem - which affects performance of the search algorithm. For our experiments, the deal sizes/lengths were based on two different distributions:

Exponential (Exp): Deals of shorter length appear more often than deals of longer lengths. For n services, there will be many deals of length less than n than of length n and the frequency of the appearance of different lengths is dictated by an exponential function. DealLength = $N.e^{(x/p)}$ where N = Total number of services, x = random number[0,1], and p = $(1/ln(1/N))$. For example, if there are 30 services and 1000 deals then about 70% of deals would have length <10, 20% would have length between 11 and 20 and the rest between 21 and 30.

Random (Ran): Length of a deal is $x.N$ where N = Total number of services, x = random number[0,1].

To answer questions a) and b) we measured the run time of the algorithm for the two distributions and results are shown in Figure 4.

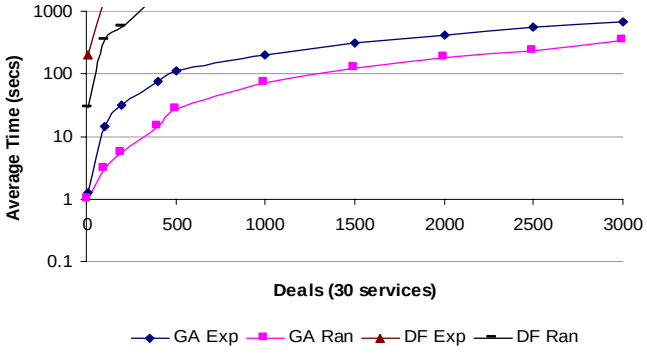

Figure 4: Run Time Comparison

The GA demonstrates excellent performance both in finding optimal solutions and as an anytime algorithm in comparison to the modified depth-first search algorithm. The depth-first search takes a much longer run time than GA whatever the deal/bid

length distribution. The difference in run time of the algorithms is considerable when the number of deals are greater. Both algorithms perform better when deal length distribution is random. The search time is greater for the exponential distribution because the search space is larger due to many shorter length deals. For each short deal, the algorithm has to try various possible deal and/or service combinations for the services not included in the short deal. These possible deal and/or service combinations is lesser when the deal length is longer. Both the GA curves in Figure 4 grow sub-linearly on the logarithmic graph, suggesting polynomial-time performance. The GA's effectiveness is strongly influenced by the distribution of deal lengths, particularly as the number of deals increase.

To answer question c) we measured the optimality of the output for both deal size distributions. Figures 5 and 6 show the anytime behavior of the GA for a particular problem. The behavior shown is similar for other problems.

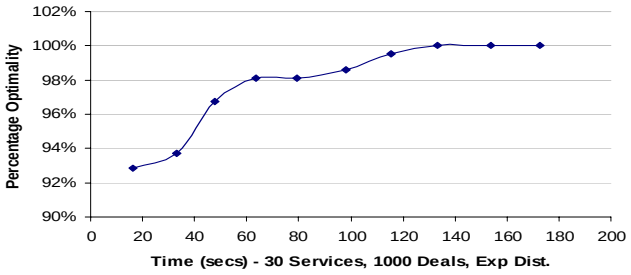

Figure 5: Anytime Behavior – Exponential Distribution

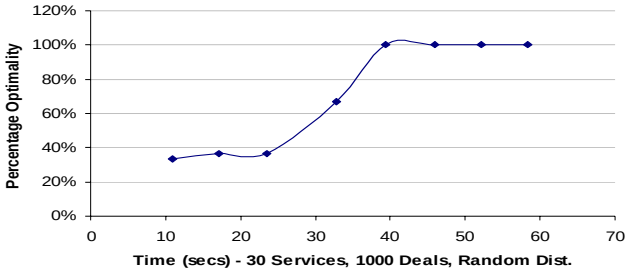

Figure 6: Anytime Behavior – Random Distribution

Good *initial* results are found quicker by the GA when deal lengths are exponentially distributed but the *final* results (i.e. optimal solution) are found quicker when the deal lengths are randomly distributed. The initial search space is greater in a random distribution as the deals are longer and finding the right length of deals to combine is difficult. The GA solves a small problem (30 services, 1000 deals) with randomly distributed deal lengths to optimality in 39 seconds whereas it takes 133 seconds when the deal lengths are exponentially distributed. The big jumps in the graph shown in Figure 6 are due to finding long deals that reduce the total cost greatly.

8 CONCLUSION AND ONGOING WORK

We presented a new brokering algorithm for optimal anytime winner determination in combinatorial auctions. Determining the winners so as to minimize a set of measures (cost, bandwidth, etc.) is NP-complete. The brokering algorithm described is a new two-stage algorithm for optimal anytime winner determination. In the first stage, a hierarchical task network planner is used to decompose a task into subtasks that can be solved by the available services. In the second stage, a genetic algorithm with heuristics is used to find the optimal combination of service providers to provide the services identified. The results from various experiments were provided. The results show the GA performs better than the modified depth-first search algorithm. The GA renders an anytime algorithm as it keeps track of best solution found from the start of the search process. We analyzed the run time and anytime performance behavior of the algorithm for exponential and random deal length distributions. The performance of the algorithm is better when the deal length distribution is random. The techniques developed to solve the brokering problem can be applied to similar NP-complete problems like the constraint satisfaction problem.

The brokering algorithm presently works on static data i.e. any changes to the data in the repository is not considered during the execution of the algorithm. The algorithm is currently modified to deal with dynamic data. We will also be applying the brokering algorithm to provide VPN (virtual private networks) service based on cost and quality of service.

ACKNOWLEDGEMENTS

We acknowledge support for this work (CASBAh project) from EPSRC, under grant GR/L34440. This project is being undertaken in collaboration with Nortel Networks and their support is gratefully appreciated.

REFERENCES

[1] P. J. Brewer and C. R. Plott. *A Binary Conflict Ascending Price (BICAP) Mechanism for the Decentralized Allocation of the Right to use Railroad Tracks*. Int. J. of Industrial Organization, 14:857-886, 1996.

[2] A. M. Easwaran and S. Drossopoulou. *A Parallel Genetic Algorithm Approach To The Knife Change Minimisation Problem*. In the Proceedings of the sixth Parallel Computing Workshop (PCW'96), Japan, November, 1996.

[3] A. M. Easwaran and J. Pitt. *A Brokering Algorithm for Cost & QoS-based Winner Determination in Combinatorial Auctions*. International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEAAIE), New Orleans, Louisiana, USA, 2000.

[4] A. M. Easwaran, J. Pitt and S. Poslad. *The Agent Service Brokering Problem As A Generalised Travelling Salesman Problem*. Autonomous Agents, Seattle, WA USA, 1999.

[5] Y. Fujishima, K. Leyton-Brown and Y. Shoham. *Taming the Computational Complexity of Combinatorial Auctions: Optimal and Approximate Approaches*. International Joint Conference on Artificial Intelligence (IJCAI), Sweden, 1999.

[6] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Publishers: Addison Wesley, 1989.

[7] E. Kutluhan. *Hierarchical Task Network Planning: Formalization, Analysis & Implementation*. PhD Thesis, Dept. of Computer Science, University Of Maryland, College Park, 1995.

[8] T. Sandholm. *An Algorithm for Optimal Winner Determination in Combinatorial Auctions*. International Joint Conference on Artificial Intelligence (IJCAI), pp. 542-547, Sweden, 1999.

[9] R. G. Smith and R. Davis. *Negotiation as a Metaphor for Distributed Problem Solving*. Artificial Intelligence,20:63-109,1983

[10] S. S. Srivastava, S. Kumar, R. C. Garg, and P. Sen. *Generalised Travelling Salesman Problem Through n Sets of Nodes.* CORS Journal, 97-101, 1969.