

Structure and Phase Transition Phenomena in the VTC Problem

Ramón Béjar	Ioannis A. Vetsikas	Carla P. Gomes	Henry Kautz	Bart Selman
Dept. of Comp. Sci. Cornell Univ. Ithaca, NY 14853 bejar@cs.cornell.edu	Dept. of Comp. Sci. Cornell Univ. Ithaca, NY 14853 vetsikas@cs.cornell.edu	Dept. of Comp. Sci. Cornell Univ. Ithaca, NY 14853 gomes@cs.cornell.edu	Dept. Comp. Sci. & Engr. Univ. Washington Seattle, WA 98195 kautz@cs.washington.edu	Dept. of Comp. Sci. Cornell Univ. Ithaca, NY 14853 selman@cs.cornell.edu

Abstract

We present a formalization of the Virtual Transportation Company (VTC) problem and study its structure and computational complexity, focusing on the job allocation component. We propose two different notions of fairness for job allocation. The problem domain has a rich underlying structure with complexity properties ranging from polynomially solvable cases to cases for which finding even approximate solutions is hard. We also give empirical results that show the existence of phase transition phenomena in the domain and suggest ways for managing the computational complexity of the allocation task in practical settings.

1 Introduction

The “Virtual Transportation Company” (VTC) problem provides a compelling framework for the study of multi-agent behaviors and their inherent computational complexity. According to the VTC problem, the Department of Defense (DOD) needs to formulate and execute plans for transporting personnel and equipment in a case of an emergency situation. A number of companies have signed agreements with the DOD to provide transportation required in such situations. The agreements involve a certain amount of regular contractual work that the DOD awards to the companies with the understanding that in case of a crisis the companies provide emergency services proportionally to the contractual work received. The companies provide estimates of what it would cost the company to provide each emergency task. After all estimates are in, the DOD goes through a job allocation process where it tries to assign jobs in a “fair” (balanced) manner to the various companies. In this paper, we focus on this job allocation process. We propose two definitions of fairness, and consider the complexity of the allocation task. The complexity of the allocation task has a number of interesting structural properties. We identify a worst-case polynomially solvable sub-case, and, in terms of average case behavior, we provide phase transition diagrams that identify the sources of complexity for the general case. In the next phase of the project, we will consider bidding strategies for the companies as well as incentives that the DOD can use to compel the companies to be truthful in their cost proposals. We also plan to

extend our complexity analysis to the full VTC domain. Such an analysis should allow us to design an efficient multi-agent bidding and allocation strategy for the VTC domain.

The paper is organized as follows. In section 2, we formalize the allocation problem. In sections 3 and 4, we present generalizations and special cases of the problem, and analyze their complexity. In section 5.1, we describe a systematic search algorithm for solving the allocation task. In section 5.2, we explain the two different models used to generate problem instances. Finally, in section 5.3, we present phase transition diagrams for the search complexity and the solution probability of the allocation problem.

2 Formalization of the allocation problem

We study the problem of assigning jobs to companies in the VTC domain. Each company has an “opportunity” cost for performing each job, which is known to that company alone. The company declares a cost for performing each job that might not be true. In this paper, we will assume that the declared costs are close to the actual costs. In the second phase of this research, we will abandon this assumption and consider incentives for getting companies to submit truthful estimates. The DOD must choose an assignment of jobs to companies based on the declared costs, such that the total cost is minimized and the total cost for each company is proportional to its income from the DOD contracts. We first consider a version of the problem where all companies have been awarded the same amount of contractual work from the DOD, so that all should share an equal total cost from implementing the crisis plan.

Minimizing the total cost leads to a solution that maximizes the social benefit, that is the total profit that all the companies make from the agreements. If this were the only goal for the department of defense, then the optimal allocation can be determined by a greedy algorithm which assigns each job to the company that has the smallest cost for that job. Under this setting, it could be the case that a certain company is assigned a large percentage of the jobs, because it can perform them for the least cost; this company would thus have a total cost much higher than the total cost for the other companies. This situation may make the company unwilling to participate in future agreements with the DOD. In related work [8], Shoham and Tennenholtz assign each job to the company that can perform it for the least cost and achieve fairness by hav-

Companies	Jobs				
	1	2	3	4	5
1	100	100	50	50	50
2	95	90	30	25	30
3	95	90	25	30	25

Table 1: Example of a cost matrix for the VTC problem. A lex min-max fair allocation is given in bold.

ing companies pay to the companies that are chosen to handle the jobs an amount of money equal to the total cost. Our approach does not involve monetary payments between companies. We explore instead strategies for obtaining a “fair” initial job allocation based on the cost matrix provided by the companies. Such an assignment needs to minimize the overall cost for performing all the jobs, while distributing the cost among companies as evenly as possible.

Let N be the number of companies, and M the number of jobs. Let $c_{i,j}$ and $c_{i,j}^D$ be the real and the declared costs for company i to handle job j . As stated above, we assume that the declared costs are approximately equal to the real ones. Table 1 gives an example of a cost matrix for the VTC problem with 3 companies and 4 jobs.

Let S_i be the set of jobs allocated to company i . A full allocation $S = \{S_1, S_2, \dots, S_N\}$ gives the job allocation for each company. The sets S_i are disjoint and their union is the set of all jobs $\{1, \dots, M\}$. Let $K_i = \sum_{j \in S_i} c_{i,j}^D$ be the total cost for company i . The goal is to select an allocation S such that the total costs K_i are fair, meaning that they are balanced, and that they are as small as possible. We consider two notions of fairness. The first definition minimizes the maximum total cost among the companies. This definition is analogous to minimizing the so-called make-span in job shop scheduling.

Definition 2.1 Min-max fairness. *A min-max fair allocation is the allocation that minimizes the maximum cost paid by any company.*

In the example in Table 1, we give a min-max fair allocation in bold face. This allocation S' assigns job # 1 to company 1, job # 2 to company 2, and jobs # 3, #4, and #5 to company 3. S' has a maximum total cost of 100 (for company 1). In this allocation, $K_1 = 100, K_2 = 90$, and $K_3 = 80$. Another min-max fair allocation, S'' would switch the assignments of jobs #1 and #2 around (*i.e.*, assign #1 to company 2, and #2 to company 1). The maximum total cost would again be 100.

Although allocations S' with S'' have the same maximum total cost, S' is preferable because it assigns a total job cost of 90 to company 2, while S'' assigns company 2 a total cost of 95. (Note that both allocations assign jobs #3, #4, and #5 to company 3.) The following notion of fairness refines the min-max fairness and gives preference of S' over S'' . With each allocation S , we introduce a cost vector $r(S)$ consisting of total costs K_i for allocation S ordered from highest to lowest. For example, $r(S') = \langle 100, 90, 55 \rangle$ and $r(S'') = \langle 100, 95, 55 \rangle$.

Definition 2.2 Lex min-max fairness. *Assignment S is*

fairer than assignment S' if and only if $r(S)$ is lexicographically smaller than $r(S')$. The lex min-max fair allocation is the allocation that has the lexicographically smallest cost vector.

This notion of fairness is analogous to the fairness concept used in load balancing in network design [4]. In the example in Table 1, S' (in bold) is the lex min-max fair allocation because its cost vector is lexicographically before that of S'' . (Also, note that other allocations are worse in terms of their cost vector. E.g., S''' with company 1 assigned $\{3, 4\}$, company 2 assigned $\{2\}$, and company 3 assigned $\{1, 5\}$ has cost vector $\langle 120, 100, 95 \rangle$.) Clearly, lex min-max fairness implies min-max fairness.

The min-max fairness optimization problem is equivalent to the *Minimum Multiprocessor Scheduling* problem, which is defined as follows. Given a set T of tasks, a number N of processors, length $l(t, i) \in \mathbb{Z}^+$ for each task $t \in T$ and processor $i \in [1 \dots N]$, find an N -processor schedule for T , *i.e.* a function $f : T \rightarrow [1 \dots N]$, with the minimum value of $\max_i \sum_{t \in T: f(t)=i} l(t, i)$. In our VTC allocation problem, if we take jobs as tasks, companies as machines, and costs as lengths of tasks, we have exactly the multi-processor scheduling problem. This problem is not only NP-hard but also not approximable within a factor of $3/2 - \epsilon$, $\forall \epsilon > 0$ in polynomial time [5].

3 Generalizations

So far, we have assumed that the income (call it A_i) that company i obtains from signing the agreement is the same for all companies. If this is not the case, then a reasonable alternative objective is to make the ratio of cost to income $\frac{K_i}{A_i}$ be as balanced as possible for all companies. An algorithm that solves the simplified (with all A_i 's identical) version of the problem can be modified to handle the more general case, by using adjusted cost values of the form $c_{i,j}^A = \frac{c_{i,j}^D}{A_i}$.

A second generalization is to consider the case where the total cost for a company does not equal the sum of the individual costs of each job assigned to it. For example, if two jobs involve each sending a crate from the same source to the same destination, then it might cost less for the same company to handle both of them compared to the sum of their individual costs. On the other hand, adding more jobs could have the opposite effect as well. The plane used for transport might reach its capacity after adding a certain number of crates, so the next crate added will mean a new flight is needed. To model this setting, one would have to allow the companies to submit more complex cost functions instead of simply a set of entries for the cost matrix. The complexity of the assignment problem would clearly increase. Characterizing such an increase in complexity is a topic for further study.

4 Boundary cases

4.1 Uniform bidding

Although in general the companies are assumed not to cooperate, it might be the case that they actually do, and, for example, give the same cost for each job. In this case, we

have $c_{i,j}^D = c_{i',j}^D \forall i, i'$. (The same value in each cell of a given column of the cost matrix.) We again want to determine whether it is possible to assign all jobs to the companies without assigning a total cost of more than K to any of them. The VTC assignment problem in this case becomes equivalent to *bin packing*. The bin packing problem is defined as follows. Given N bins of size $B > 0$ and a finite set of items U with each item $u \in U$ having a size $s(u) \in \mathbb{Z}^+$ is it possible to partition the set U into disjoint subsets U_i , such that $\max_i \sum_{u \in U_i} s(u) \leq B$? The bin-packing problem is NP-complete but good approximation schemes and average-case results are known.

4.2 Uniform cost

Another special case is when each company declares the same cost for each job it can handle, that is $c_{i,j}^D = c_{i,j'}^D = c_i, \forall j, j'$. This can happen when the various tasks are all quite similar (e.g., the transport of similar units). In the cost matrix, we would have the same value in each cell of a given row. In this case, there is a polynomial time algorithm that gives the lex min-max fair allocation.

```

set  $k_i = 0, \forall i = 1, \dots, N$ 
for  $j = 1, \dots, M$ 
{ find  $S^I = \operatorname{argmin}_{i'} \{k_{i'} + c_{i'}\}$  and
 $i = \operatorname{argmax}_{i'' \in S^I} \{k_{i''}\}$ , then set  $k_i = k_i + c_i$  }

```

Note that S^I is a set since we want all the i'' 's that minimize the argument, but for the *argmax* we can take any of them at random, if there are more than one i'' 's that maximize the argument. In the end, k_i is the cost for company i and $\frac{k_i}{c_i}$ is the number of jobs that should be assigned to it. This algorithm runs in time $O(M \times N)$. It can be shown that this algorithm finds a lex min-max fair allocation.

The algorithm also works in a more general setting. Assume that each company i has a limit λ_i , which is the maximum number of jobs that can be assigned to that company. This generalization is based on the fact that a company may not have the resources to handle too many jobs. The original algorithm can be modified so that, if a company has reached its limit $k_i = \lambda_i \times c_i$, it is not chosen to participate in set S^I anymore.

5 Characterizing the complexity of the allocation problem

The complexity of the VTC allocation problem is determined by the form of the cost matrix. As noted above, the general problem is NP-complete, but one can obtain useful further insights into the complexity of the problem by considering certain special cases for the cost matrix. In the previous section, we considered two boundary cases. The uniform cost scenario leads to a polynomial time solvable problem, and the uniform bidding scenario leads to a problem equivalent to the knapsack problem, which is NP-complete but for which good approximation methods are known. In practice, the form of the cost matrix will lie in between these extremes. To study

the general scenario, we will now present data on the hardness of the min-max fair VTC allocation problem under two different instance distributions. For related work on the interplay between structure and average-case complexity, see [1; 2; 7].

5.1 A decision algorithm for min-max fair allocation

We use a backtrack search method to determine whether an allocation exists given a predefined maximum cost K for the companies. The algorithm starts the search with no jobs assigned to any company and with a total accumulated cost for every company set to zero. At every node of the search tree, the algorithm selects an unassigned job and assigns this job to one of the companies that can perform the job without increasing its accumulated cost above the upper limit K . When the algorithm reaches a point where for some unassigned job there are no companies available, it backtracks to reconsider its previous assignments.

A branching heuristic and a value ordering heuristic are used to try to find a solution as quickly as possible. At every node, we pick the job with the minimum number of companies able to perform it and we consider the available companies in decreasing order of their accumulated costs (provided the costs are below the limit K). The intuition behind these heuristics is to try to branch first on a variable that produces a subtree with a minimal branching factor, and, in the value selection, we try to find an inconsistent partial solution as soon as possible to prune the search space.

5.2 Instance distributions

In order to study average-case complexity properties of our domain, we need to decide on a problem instance distribution. We will consider two models for generating the instances. In each model, we first need to specify the number of jobs M , the number of companies N , and a closed interval of allowed integer costs for the jobs ($[LC, UC]$). We then decide, for every job, which companies can perform the job. Finally, the cost of performing a job for a chosen company is selected uniformly at random from the interval $[LC, UC]$.

The models differ in the way jobs are selected for each company. In the *fixed connectivity model*, we use an additional parameter c . For each job, we select uniformly at random, c companies that can perform the job. In the *constant-probability model*, we use instead a fixed probability p , with $0 < p \leq 1$ for selecting companies. For each job, each company is selected for the job with probability p . These instance distributions are motivated by the two most commonly studied probability distributions for the Boolean satisfiability problem [6].

5.3 Phase transition phenomena and complexity

We now present our experimental data for the average-case complexity of finding a min-max fair allocation with a maximum cost less than a given limit K . For each of our random instance distributions, we generate instances with a fixed number of jobs (45) and a fixed interval of allowed costs ($[40, 100]$), but we vary the number of companies among the instances to study the effect of having less companies than

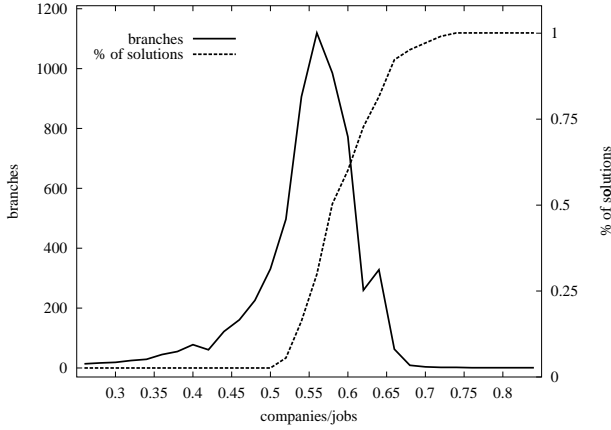


Figure 1: Complexity and phase transition for fixed connectivity model with $c = 3$.

jobs, almost the same number of companies as jobs, and more companies than jobs. When solving the instances the upper limit cost (K) has been fixed at 131. In our setting, this resulted in allocations with a relatively low probability of having a company perform more than two jobs. In future work, we will study the influence of the choice of the value of K on the overall results.

For the fixed connectivity model, we considered instances for several values of c . Figure 1 gives the data for the $c = 3$ case, which is representative of the average-case complexity as a function of the ratio of the number of companies to the number of jobs (x-axis). The solid line shows the median number of branches in the search tree visited by the algorithm. The dashed line shows the percentage of instances that were found to have a solution. We observe that for instances with few companies compared to the number of jobs, it is relatively easy to determine that there is no fair allocation with K less than the preset limit. Increasing the number of companies, increases the chance of finding solvable problem instances but also the complexity of solving the allocation task. When we continue to increase the number of companies, the median search cost begins to decrease, leading to another region where the instances are easily solved. So, the ratio of companies to jobs provides the right parameterization for the problem and we obtain the characteristic easy-hard-easy pattern in the computational difficulty of solving the instances [6; 3]. Looking at the dashed curve, we observe that the easy region on the left corresponds to instances that almost always have no solution, the easy region of the right corresponds to instances that almost always have a solution. In between these two regions, we observe a sharp transition from unsolvable instances to solvable instances. The point where 50% of the instances have a solution corresponds to the peak in the search complexity.

Figure 2 shows the curves for the percentage of solvable instances for three values of c . We observe that when increasing c the location of the threshold point decreases and the transition becomes sharper. A possible explanation for the change in location is that by increasing c , while keeping the

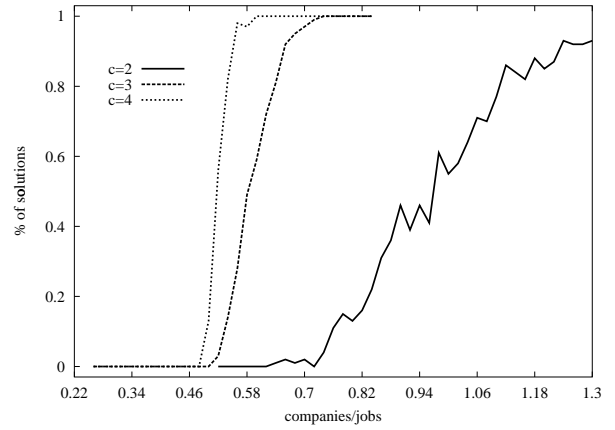


Figure 2: Phase transition for fixed connectivity model with $c = 2, 3$ and 4 .

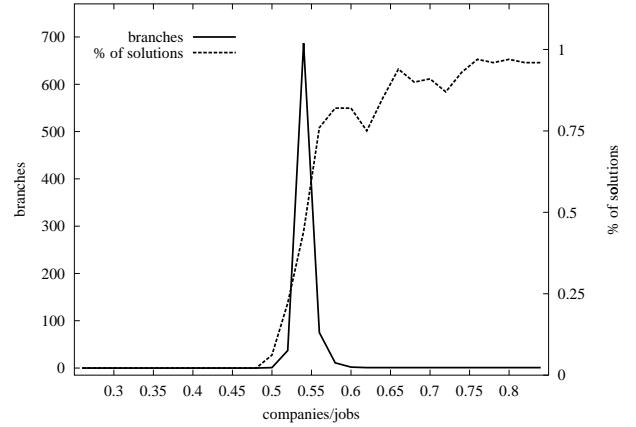


Figure 3: Complexity and phase transition for the constant probability model for $p = 0.18$.

number of jobs and companies fixed, we increase the probability that there exists an allocation for all the jobs, since each job can now be performed by more companies and, given our fixed connectivity model for assigning jobs, it is more probable that one or more of these companies can perform the job for a low cost. Therefore, we reach the solvable region with a smaller number of companies. Concerning the difficulty of solving the instances, we observe that at the phase transition threshold, the median number of branches is about 4 for $c = 2$, 1000 for $c = 3$, and $1 \cdot 10^6$ for $c = 4$. So, we have an apparent exponential scaling in c . This is consistent with findings for the K-SAT model, for increasing values of K .

For the constant-probability model, we consider instances obtained with three different values of p : 0.16, 0.17 and 0.18. We illustrate the general behavior observed in the three cases by focusing on the $p = 0.18$ case. See Figure 3. We observe a similar behavior as for the fixed connectivity model. However, in this case, we still find unsolvable instances far away to the right of the threshold. This is due to the fact that

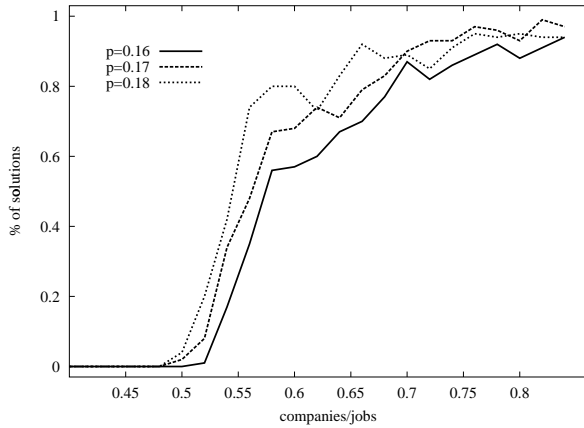


Figure 4: Phase transition for the constant probability model with $p = 0.16, 0.17$ and 0.18 .

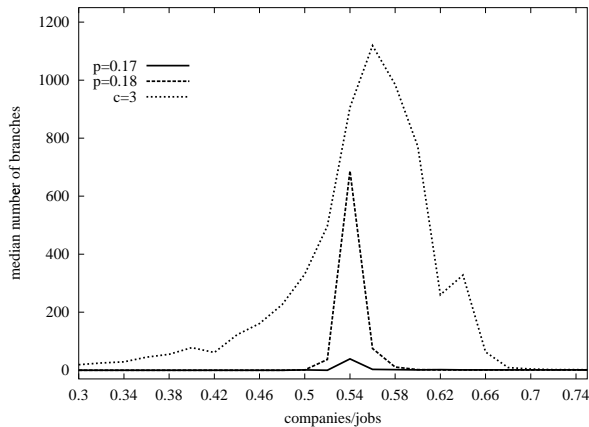


Figure 5: Comparing the difficulty of solving instances from the two instance distributions.

in this model, there is a finite probability that some jobs are not assigned to any companies. Figure 4 shows the curves for the percentage of solvable instances for the three cases of p . We observe a similar behavior as with the fixed connectivity model. However, this time the values for the median number of branches in the threshold point are much lower, 2 for $p = 0.16$, 3 for $p = 0.17$, and 687 for $p = 0.18$.

In Figure 5, we compare the relative hardness of the constant-probability model and the fixed-connectivity model. A value of $p = 0.18$ corresponds to an average of 4.4 companies per job. For our fixed-connectivity model, we used a connectivity of $c = 3$ companies per job. (With $c = 4$, the curve would lie still higher.) The figure shows that the fixed-connectivity model leads to substantially harder instances than the fixed-probability model. These findings are consistent with what happens in the two analogous random models for the SAT problem, the random K-SAT model and the constant-probability model [6].

6 Conclusions and future work

We have presented a formalization of the job allocation component of the VTC problem with two alternative definitions for the notion of fairness. We also presented an initial analysis of the computational difficulty of finding fair assignments. The complexity of this task is determined by the structure of the cost matrix. The ratio of companies to jobs provides a natural parameterization of the problem, and leads to phase transition phenomena in which hard regions of the problem space can be identified.

In the next phase of the project, we will consider the bidding process for companies for submitting their costs for performing the various tasks. In particular, we will study mechanisms that can be used to compel a company to submit a cost figure that is as close as possible to the true cost for the company. We will also consider the complexity of the overall bidding process. Combined with our findings for the complexity of the job allocation process, a detailed understanding of all sources of complexity should enable us to design an overall efficient and fair multi-agent approach for the VTC problem.

References

- [1] D. Achlioptas, C. P. Gomes, H. Kautz, Y. Ruan, B. Selman, and M. Stickel. Balance and Filtering in Structured Satisfiable Problems. In *To Appear in: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, Seattle, 2001.
- [2] C. P. Gomes and B. Selman. Problem Structure in the Presence of Perturbations. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, New Providence, RI, 1997. AAAI Press.
- [3] S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264:1297–1301, 1994.
- [4] J. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. In *FOCS: 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1999.
- [5] J. K. Lenstra, D. B. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming*, 46:259–271, 1990.
- [6] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the 10th National Conference on Artificial Intelligence, AAAI'92, San Jose/CA, USA*, pages 459–465. AAAI Press, 1992.
- [7] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic ‘phase transitions’. *Nature*, 400:133–137, 1999.
- [8] Y. Shoham and M. Tennenholtz. The fair imposition of tasks in multi-agent systems. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'01 (to appear)*, 2001.