Asynchronous Teams: a Multi-Algorithm Approach for Solving Combinatorial Optimization Problems

Rosiane de Freitas Rodrigues *Pedro Sérgio de Souza*

Relatório Técnico DCC-95-18

Novembro de 1995

Asynchronous Teams: a Multi-Algorithm Approach for Solving Combinatorial Multi-Objective Optimization Problems[1]

Rosiane de Freitas Rodrigues[2]
Pedro Sérgio de Souza

Departamento de Ciência da Computação
Instituto de Matemática, Estatística e Ciência da Computação
Universidade Estadual de Campinas
CEP: 13081.970
Campinas - SP Brasil
e-mail: {rosiane | pss}@dcc.unicamp.br
Phone: +55 192 39-8442/3115 FAX: +55 192 39-7470/5808

Abstract

The fundamental question in solving multi-objective function problems lays in the determination of solutions that would best meet all the objectives involved. The aim of this work is to present Asynchronous Teams (or A-Teams) as an appropriate method to detect this set of solutions for combinatorial problems. A-Teams basic principle is the asynchronous cooperation among a set of heuristic algorithms in order to produce better solutions than those obtained using each algorithm separately. As an example of a combinatorial multi-objective function problem we propose the Traveling Salesman Problem with various distance matrices.

Key words: vector optimization, combinatorial algorithms, non-dominated solution set.

1. Introduction

Many problems of practical interest are multi-objective [HPY80]. Usually, these objectives conflict with each other in such a way that an objective cannot be improved without causing degradation in some of the others - as, for example, the minimizing of costs and the maximizing of service quality in a company.

The presence of conflicting objectives hinders the existence of an optimal solution for the given problem, which leads us to look for the best compromise solutions considering all the objectives involved [Zel82]. And this makes the task even more complex than the simple seeking for an optimal [Mur92].

[1] This work was presented at the 5th Workshop of the DGOR-Working Group: Multicriteria Optimization and Decision Theory, March 1995. Supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES and by Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP.

[2] Linked to the Federal University of Amazonas.

This article is elaborated upon a multi-algorithm method yielding alternative multiple solutions, the Asynchronous Teams or A-Team [Sou93], whose efficiency has already been shown to classic problems with a single objective function in recent research [Sou93] [Mur92].

Mathematically, problems with multi-objective functions can be expressed in this way [HPY80]:

$$min\ f(x) = \{f_1(x), f_2(x), \ldots, f_k(x)\} \qquad s.t.\ \ g_i(x) \geq 0\ \ ,\ \ i = 1,\ \ldots, m \tag{1}$$

where x is a n dimensional vector decision variable, pertaining to R^n and the functions $f(x)$ and $g_i(x)$ can be either linear or non-linear functions. The problem thus consists of n decision variables, m restrictions and k objectives ($k \geq 2$).

Such problems are known as vector minimization problems, where minimize means to obtain solutions to meet all the problem objectives in the best way. We now have enough elements to introduce some fundamental concepts:

Definition 1: ***Dominance***. Given $f_i(\cdot)$, $i=1,2,\ldots,k$ objective functions to be analyzed, if $f_i(x_b) \leq f_i(x_a)$ for every i and there exists $j, j \in \{1,\ldots,k\}$, such that $f_j(x_b) < f_j(x_a)$, then x_b dominates x_a.

As mentioned before, all the objectives conflict with each other, so an optimal solution able to minimize simultaneously all the objective functions is almost impossible to be obtained.

Definition 2: ***Non-dominated solutions (Pareto Optimal)***. A vector $x_e \in R^n$ is a non-dominated solution of the above problem if $g_i(x_e) \geq 0$ for all $i \in \{1,\ldots,m\}$ and if there does not exist $x \in R^n$ with $g_i(x) \geq 0$ for all $i \in \{1,\ldots,m\}$ which dominates x_e with respect to $f_j(x)$, $j \in \{1,\ldots,k\}$.

This paper aims to show the adequacy of Asynchronous Teams as an adequate method to detect solutions pertaining to the Pareto optimal or as close as possible when solving combinatorial problems with multi-objective functions. It also proposes a generalization of the classical Traveling Salesman Problem for several distance matrices, named Multi-Distance Traveling Salesman Problem - MDTSP.

2. Multi-Distance Traveling Salesman Problem

A traveling salesman wants to visit a whole set of cities one at a time and finish his visit in the very city where he started from. Furthermore, he has to minimize **all** *the costs involved in his journey.*

This is the Multi-Distance Traveling Salesman Problem - MDTSP [ST93], proposed as a generalization of the NP-Hard Traveling Salesman Problem - TSP [GJ79] [Law85] where instead of only one, various distance matrices must be handled simultaneously. We then deal with a combinatorial multi-objective function optimization problem, which can be structured on the basis of integer programming [DFJ54]:

Definition 3: Integer variables x_{ij} indicate whether or not a city i is visited after a city j, respectively (x_{ij} = 1) or (x_{ij} = 0). The costs of going from city i to city j are given by c_{ij}^{d}. So:

$$min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{d} x_{ij} \qquad , \quad d = 1, \ldots, k \tag{2}$$

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad , \quad j = 1, \ldots, n \tag{3}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad , \quad i = 1, \ldots, n \tag{4}$$

$$\sum_{i \in S}\sum_{j \in S} x_{ij} \leq |S| - 1 \tag{5}$$

where, exactly as in the original problem, n is the number of cities of the problem, S is a nonempty, proper subset of the set $\{1, 2, \ldots, n\}$, the symbol "|●|" denotes the set cardinality and k is the *problem number of objective functions*, in our case the number of distance matrices. Formula (2) is the problem general representation, indicating that we are willing to minimize the summation of distances among all the connections between the existing cities, taking into account the k distance matrices. As for (3), (4) and (5), they represent the restrictions that have to be respected, or rather, equations (3) and (4) guarantee that every city must have a connection coming from one city and another connection going to another city, and the inequality (5) guarantees that no subtour can be created.

The broadly proven suitability of numberless problems modeling for a TSP and the detection of other practical problems that can be modeled for a MDTSP, motivated its elaboration. For example, scheduling problems with one inter-task cost, where the order according to which tasks are scheduled shows distinct costs, when the task scheduling sequence is cyclic (and if it is not, we just have to create a connection with cost zero between the first and all the other tasks in the sequence), we do have a typical TSP, i.e., we are looking for the sequence that goes through all the tasks with minimum cost, without repeating any one of them. In the case of a scheduling problem where we have several inter-task costs and need to consider them all to obtain the best task scheduling sequence (for example, time minimization and quality maximization), we come to the Multi-Distance Traveling Salesman Problem.

There is a class of methods for the resolution of TSP whose algorithms may be modified to solve MDTSP: it is the class of approximate methods. The approximate methods (heuristics or meta-heuristics) do not always guarantee the finding of an optimal solution, but, usually, provide almost optimal solutions in a reasonable execution time, besides being, most of the times, much easier to implement than exact methods.

The A-Team method is a kind of meta-heuristic that involves several heuristics, some of which are even considered useless separately, and allows the algorithms to help each other in an asynchronous way, sharing the same data, and yielding very good solutions. This method will be discussed now.

3. Asynchronous Teams

Asynchronous Teams or A-teams consist in an organization of autonomous agents that communicate asynchronously through shared memories, and contain a cyclic data flow. In such a way, algorithms can yield solutions that can be shared by all, leading to optimal or close to optimal solutions [Sou93]. A-Team's main characteristics are:

Autonomous agents → Set of independent algorithms, so that new agents may be inserted to and old ones removed from the set without affecting the performance of the others.

Asynchronous communications → The agents manipulate data in shared memories without any synchronization between them , which allows them to be executed in parallel or concurrently.

Cyclic data flow → There must be at least one data flow cycle, in order to allow a continuous iteration flow among all the agents that are going to manipulate - read, modify and store - information in the shared memory.

The link among the algorithms is the data structure used in the shared memories. No matter what an algorithm executes, its output has to be appropriately formatted to be stored in the memory which is assigned to receive the outcoming data of the algorithm. The same rationale applies for the input memories: an algorithm must read a specific data format from each input memory that it accesses.

Each solution written by an algorithm in a shared memory is stored in one of its many slots. Several algorithms are free to access any of those slots at any time. The only restriction that applies for consistency proposes is to follow a *semaphore mechanism*, so that each slot is considered as a *"critic region of memory"* [Tan92]. That is, if one algorithm is writing in a slot, nobody else can either read from or write to the slot. No other protocol than that is required for algorithm communication.

Souza [Sou93] has shown that A-Teams for the Traveling Salesman Problem have better performance than each algorithm can archive alone. He also has shown that the more algorithms an A-Team has, the better its performance is. Furthermore, he has run experiments with several machines and obtained results close to ideal situations: execution time for one machine was divided by 2, 3 and 4 when using 2, 3 and 4 processors, respectively. These results have motivated the idea to expand his work for multi-objective function problems.

3.1. A-Teams for the MDTSP

The A-Team method generates multiple solution alternatives, close to optimal or even optimal solutions for a given single objective combinatorial problem. In the case of multi-objective function problem resolution, *finding the optimal solution* does not apply. The goal is to *find the set of the most efficient solutions* for the *n* objectives involved.

The elaboration of an A-Team is a relatively simple task. Firstly, we have to gather the available algorithms for the resolution of the problem to be faced and create a data flow to connect these algorithms. In our case, there is no algorithm available since the MDTSP has just been defined. It is necessary to modify some algorithms for the TSP to generate solutions with relation to *all distance matrices* involved. For the sake of simplicity regarding the computational testing, we have implemented an A-Team to solve a

MDTSP in two dimensions (only two distance matrices), but this does not implicate in lose of generality, since except for computational complexity, nothing else depends on the number of dimensions involved.

The memories of an A-Team need to be initialized with an adequate size and a high diversity of solutions. We decided to initialize the memory with a number of tours equal to the instance size, and to adopt the maximum size of the complete tours memory with the double of the instance size.

3.1.1. The algorithms used in the A-Teams

Five algorithms for the classic Traveling Salesman Problem (TSP) were chosen.

The algorithm ***DEC*** belongs to the class firstly defined by Souza [Sou93] specifically for the TSP: ***tour deconstruction algorithms***. DEC uses the *common edges* of two tours to generate a partial closed tour that will be completed by the Furthest Insertion algorithm, defined below.

The ***Furthest Insertion*** (***FI***) algorithm [Law85] belongs to the class of heuristic algorithms called ***tour construction algorithms***. At each step, this algorithm inserts a city to partial solutions, until a complete solution is formed.

The other three algorithms belong to the class of heuristics known by ***tour improvement heuristics*** [Law85], that are: ***Lin-Kernighan*** (***LK***) [LK73], ***2-opt*** [Ben90] and ***3-opt*** [SDK83]. Given a feasible tour (valid solution) generated by any means, in our case for FI, these algorithms try to improve this tour in different ways, generating *new tours*, until no shorter tour is obtained. Tour improvement algorithms are also known as local search algorithms [Law85].

While searching for better solutions with respect to one of the objective functions, improvement algorithms find intermediate solutions, that can be incorporated as part of the best (non-dominated) solutions found.

Besides the available algorithms, size and initialization of memories, there is another important parameter for designing an A-Team that will be discussed bellow.

3.1.2. Internal structure of the memory of valid solutions

The internal structure of the memory of valid solutions is one of the crucial points in the developing of an Asynchronous Team. Considering that the size of this memory cannot be unbounded, we introduced a ***Destroyer algorithm*** *(* D_1 *)* to keep the size of the memory constant. It keeps the solution ordered by increasing value of the following function:

$$F(x_1, x_2, \ldots, x_n) = \sqrt{\sum_{i=1}^{k} f_i^2(x_1, x_2, \ldots, x_n)} \tag{6}$$

where k represents the number of objectives of the problem (fi (●)) and n represents the number of variables of the problem. Once a newcoming solution *x'* is sent to this memory and it has a value *f(x')* smaller than the worst solution in memory, the destruction algorithm eliminates from the memory the worst solution (the most distant solution from the origin of the axes) to open space for the new solution.

The destruction policy adopted by the destroyer algorithm eliminate from the memory the most distant solutions from the origin of the axes (objectives) , that is, the solutions with higher values in the memory, since the memory contains the scalar objective function solutions in ascending order.

4. Results

The results were obtained for five instances of the Multi-Distance Traveling Salesman Problem with 9, 50, 100, 200 and 500 cities. For each instance, two distance matrices were randomly generated with integer numbers in the interval from 1 to 100.

The 9-city instance was applied to 4 configurations of A-Teams. Configuration 1 is composed by algorithm ***DEC*** (***deconstructor***), ***FI*** (***Furthest Insertion***), and the ***destroyers***. Configuration 2 adds the ***2-opt*** algorithm, configuration 3 adds the ***3-opt*** algorithm and configuration 4 adds the ***Lin-Kernighan*** algorithm (Fig. 4.1). All other instances were tested with configuration 4 only.

In this figure (Fig. 4.1), the data flow of the A-Team starts with a full initialization of the memory (rectangles) with tours. FI_1 and FI_2 initialize the memory of valid solutions. FI_1 is a version of the FI that uses the first distance matrix to generate tours (50% of the memory) and FI_2 uses the second distance matrix. All other algorithms are indexed in the way with respect to the usage of matrix 1 and 2. Note that algorithms are represented by arrows.

After initialization, all the algorithms of the A-Team (*LK*, *2-opt*, *3-opt*, *FI*, *DEC* and the *Destroyers*) start running. The LK and 2-opt read and write solutions in the memory of valid solutions. The DEC reads from memory of valid solutions and writes in the memory of partial solutions. The FI reads from memory of partial solutions and writes in the memory of coarse solutions. The 3-opt reads from memory of coarse solutions and writes in the memory of valid solutions. All this is processed asynchronously.

The Destroyer D_1 eliminates the worst tour in the memory of valid solutions every time that a new solution is add to the memory. D_2 and D_3 adopt the policy *First In First Out* to keep the size of the memories of partial solutions and coarse solutions constant.

The A-Team stops after an arbitrary number of algorithm executions that depends on the instance size. The order that the algorithms executed was randomly chosen.

Fig. 4.2 shows the number of solutions that belongs to the Pareto optimal for the 9-city instance for each configuration. The more algorithms the configuration has, the better the set of non-dominated solutions that is found. Figs. 4.3, 4.4, 4.5, 4.6 and 4.7 show these sets found for instances of 9, 50, 100, 200 and 500 cities using configuration 4. The two sets of points close to the high end of the axis represent the initial solutions provided by the Furthest Insertion algorithm. Each axes represents the tour length of a solution with respect to the two distance matrices. For all of them, the A-Team was able to found a set of non-dominated solutions spread out along the axes.

5. Conclusion

As we can see from the results, A-Teams can generate large sets of non-dominated solutions for all the instances of the MDTSP tested. We conjecture that A-Teams can be applied to a large number of combinatorial multi-objective function problems to generate near-optimal sets of non-dominated solutions.

Acknowledgment

The authors would like to thank Dr. Sarosh Talukdar (Carnegie Mellon University) and Dr. Seshashayee Murthy (T. J. Watson Research Center, IBM) for the discussions on the basic ideas of this work.

Bibliography

[Ben90] BENTLEY, J. L. Experiments on traveling salesman heuristics, Proc. 1st. Ann. ACM-SIAM Symposium on Discrete Algorithms, SIAM. Philadelphia, 1990.

[DFJ54] DANTZIG, G. B., FULKERSON, D. R., JOHNSON, S. M. *Solution of a large-scale traveling salesman problem,* Operations Research, 1954.

[GJ79] GAREY, Michael R., JOHNSON, David. *Computers and Intractability*; A Guide to NP-Completeness. San Francisco: W. H. Freeman, 1979.

[HPY80] HWANG, C. L., PAIDY, S. R., YOON, K. Mathematical Programming with Multiple Objectives; A Tutorial. Computer & Operational Research, Great Britain: Pergamon Press, 1980.

[Law85] LAWLER, E. L. et al, (Ed.). *The Traveling Salesman Problem*. Chichester: John Wiley & Sons, 1985.

[LK73] LIN, S. KERNIGHAN, B. W. *An Effective Heuristic Algorithm for the Traveling Salesman Problem,* Operations Research, 1973.

[Mur92] MURTHY, Seshashayee S. *Synergy in Cooperating Agents; Designing Manipulators from Task Specifications*. Ph. D. dissertation, Electrical and Computer Engineering Department, Pittsburgh: Carnegie Mellon University, 1992.

[SDK83] SYSLO, M. M., DEO, N., KOWALIK, J. S. *Discrete Optimization Algorithms with Pascal Programs*. Englewood Cliffs: Prentice Hall, 1983.

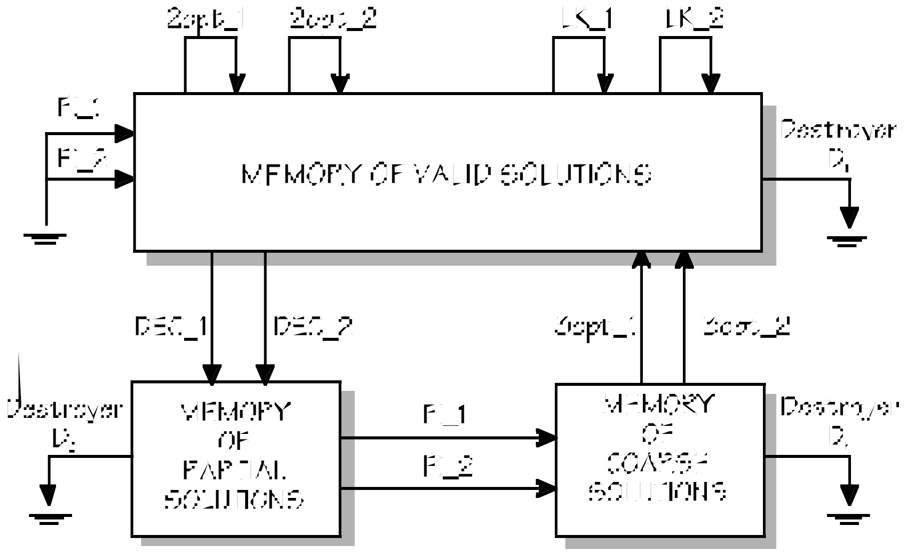
[SR91] SHIN, W. S., RAVINDRAN, A. *An Interactive Method for Multiple-Objective Mathematical Programming Problems*. Journal of Optimization Theory and Applications, Plenum, 1991.

[Sou93] SOUZA, Pedro Sérgio de. *Asynchronous Organization for Multi-Algorithm Problems*. Ph. D. dissertation, Electrical and Computer Engineering Department, Pittsburgh: Carnegie Mellon University, 1993.

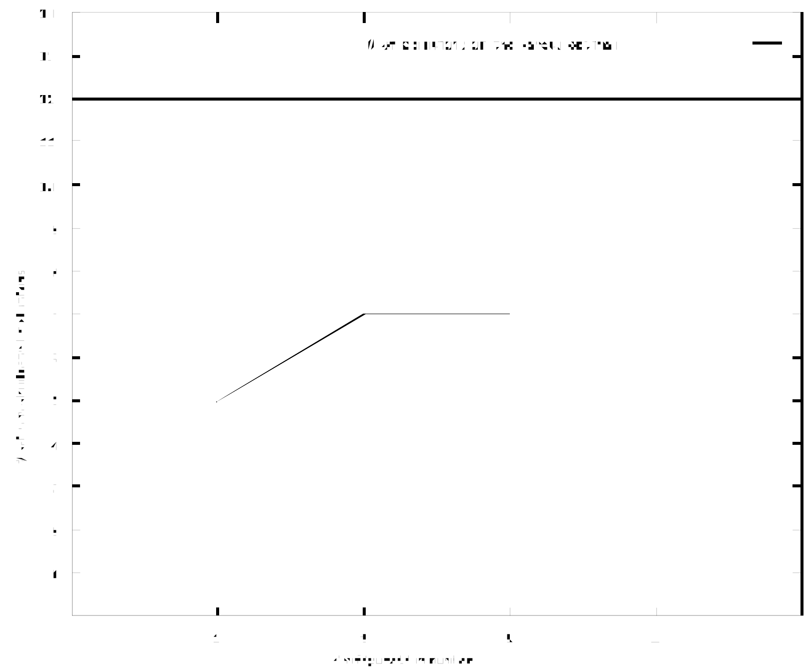
[ST93] SOUZA, Pedro Sérgio de, TALUKDAR, Sarosh N. *Personal communication.*

[Tan92] TANENBAUM, Andrew S. *Modern Operating Systems.* New Jersey: Prentice-Hall, 1992.

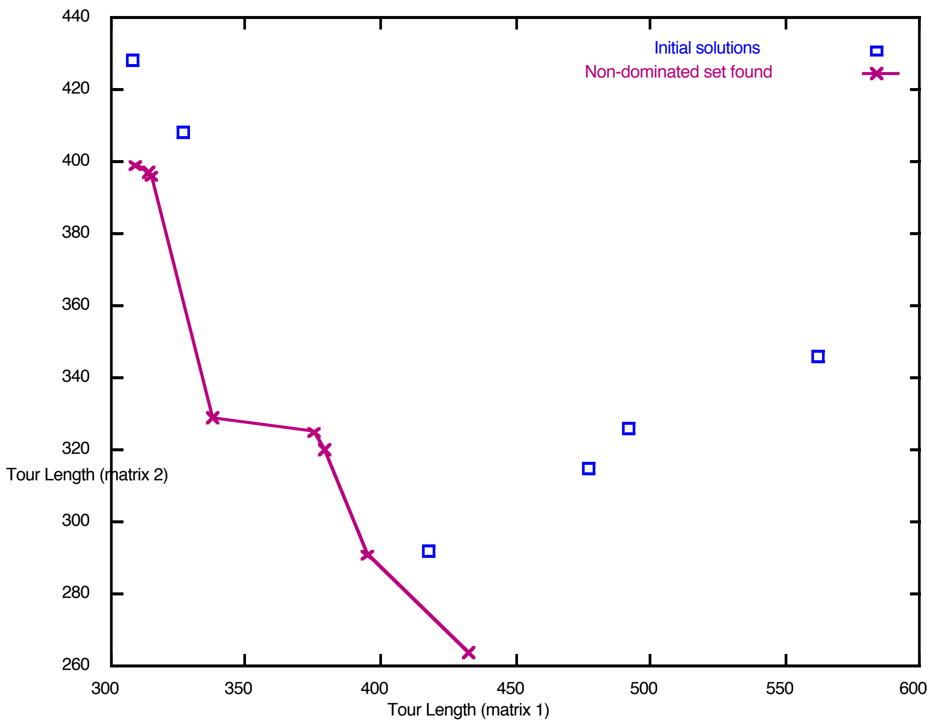
[Zel82] ZELENY, M. *Multiple Criteria Decision Making.* New York: McGraw-Hill, 1982.



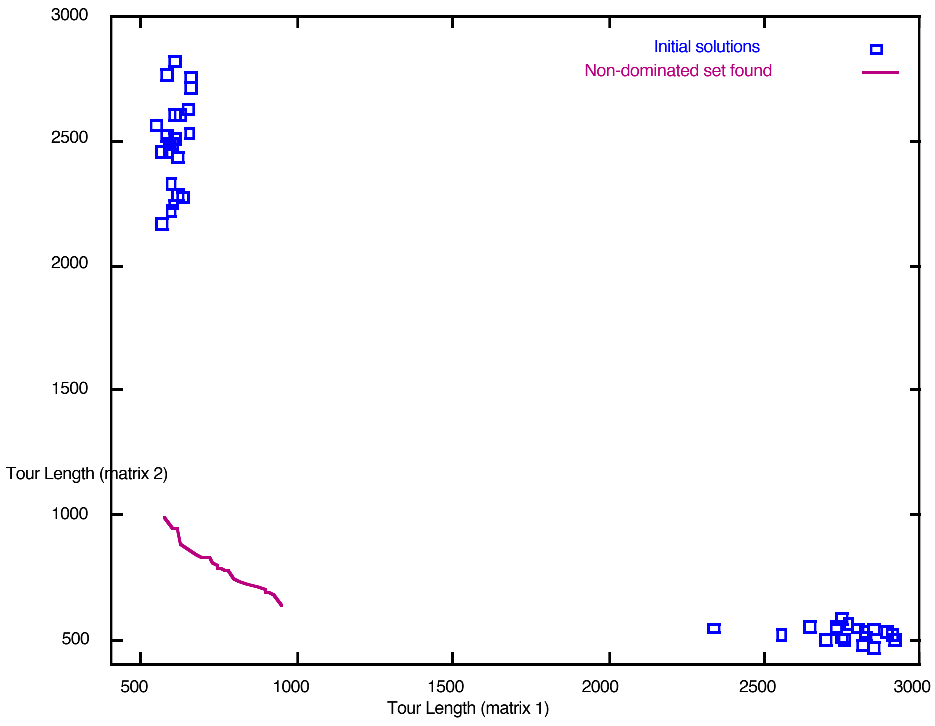
4.1. Configuration 4 (the most complete) of the A-Teams elaborated.



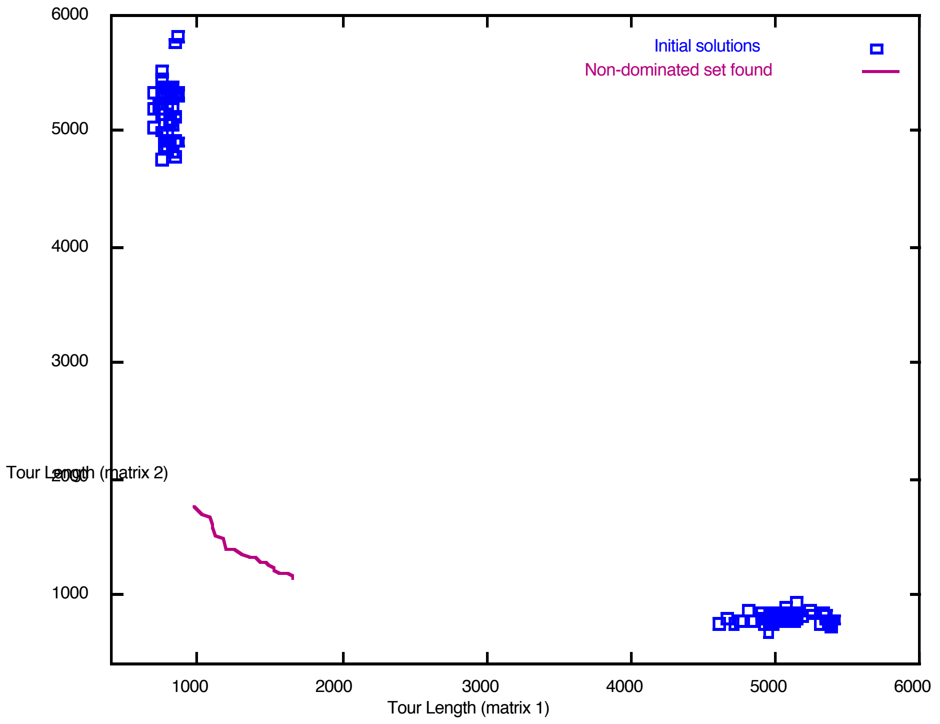
4.2. Comparison of the number of non-dominated solutions generate of A-Team in each one of the configurations, for the 9-city instance.



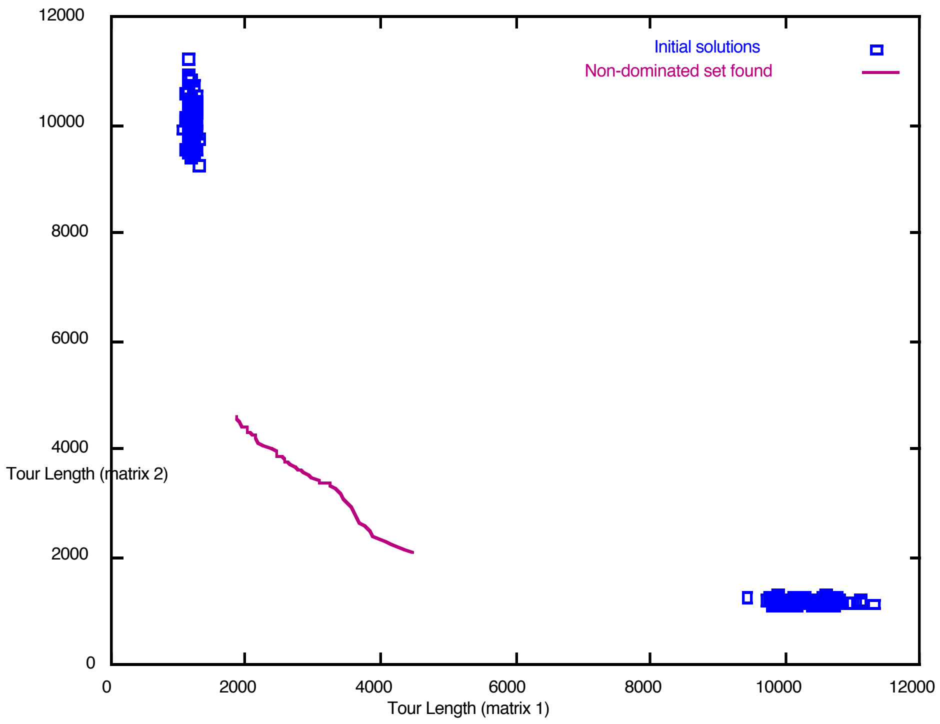
4.3. The best set of non-dominated solutions generated by the A-Team for the 9-city instance.



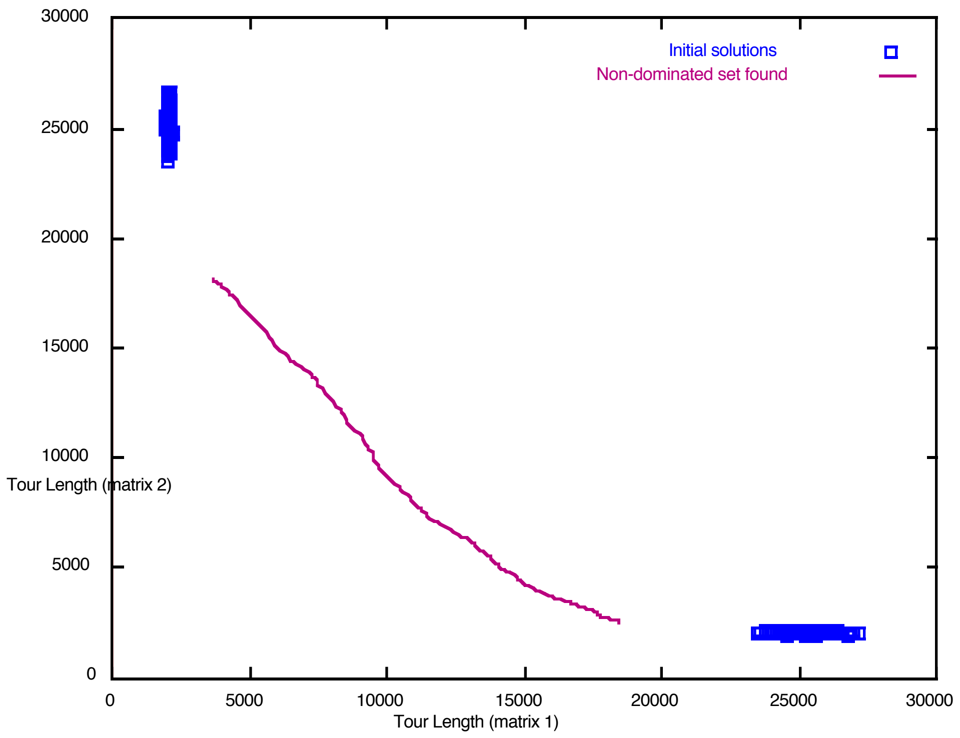
4.4. The best set of non-dominated solutions generated by the A-Team for the 50-city instance.



4.5. The best set of non-dominated solutions generated by the A-Team for the 100-city instance.



4.6. The best set of non-dominated solutions generated by the A-Team for the 200-city instance.



4.7. The best set of non-dominated solutions generated by the A-Team for the 500-city instance.