Hindawi
Complexity
Volume 2017, Article ID 3813912, 12 pages
https://doi.org/10.1155/2017/3813912

WILEY | Hindawi

Research Article

The Multiagent Planning Problem

Tamás Kalmár-Nagy,[1] Giovanni Giardini,[2] and Bendegúz Dezső Bak[1]

[1]*Department of Fluid Mechanics, Faculty of Mechanical Engineering, Budapest University of Technology and Economics, Budapest, Hungary*
[2]*SPS Italiana Pack Systems, Novara, Italy*

Correspondence should be addressed to Tamás Kalmár-Nagy; complexity@kalmarnagy.com

Received 31 July 2016; Revised 17 December 2016; Accepted 4 January 2017; Published 5 February 2017

Academic Editor: Roberto Natella

Copyright © 2017 Tamás Kalmár-Nagy et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The classical Multiple Traveling Salesmen Problem is a well-studied optimization problem. Given a set of n goals/targets and m agents, the objective is to find m round trips, such that each target is visited only once and by only one agent, and the total distance of these round trips is minimal. In this paper we describe the Multiagent Planning Problem, a variant of the classical Multiple Traveling Salesmen Problem: given a set of n goals/targets and a team of m agents, m subtours (simple paths) are sought such that each target is visited only once and by only one agent. We optimize for minimum time rather than minimum total distance; therefore the objective is to find the Team Plan in which the longest subtour is as short as possible (a min–max problem). We propose an easy to implement Genetic Algorithm Inspired Descent (GAID) method which evolves a set of subtours using genetic operators. We benchmarked GAID against other evolutionary algorithms and heuristics. GAID outperformed the Ant Colony Optimization and the Modified Genetic Algorithm. Even though the heuristics specifically developed for Multiple Traveling Salesmen Problem (e.g., k-split, bisection) outperformed GAID, these methods cannot solve the Multiagent Planning Problem. GAID proved to be much better than an open-source Matlab Multiple Traveling Salesmen Problem solver.

1. Introduction

Applications from space exploration [1–3] and drone delivery to search and rescue problems [4–7] have underlined the need to plan a coordinated strategy for a team of vehicles to visit targets. It is important to develop a multiagent planner for a team of autonomous vehicles to cooperatively explore their environment [8, 9]. We formulate the overall planning problem as finding a near-optimal set of paths that allows the team of agents to visit a given number of targets in the shortest amount of time. This problem is quite similar to the well-known Multiple Traveling Salesmen Problem (MTSP) [10–13], a generalization of the Traveling Salesman Problem (TSP) [14–16], that can be stated as follows: given n nodes (targets) and m salesmen (agents), the MTSP consists of finding m closed tours (paths start and end at the starting point of the agents), such that each target is visited only once and by only one agent and the total cost of visiting all nodes is minimal. MTSP has been used for modeling many real situations, from scheduling activities of companies and industries to cooperative planning problems. See, for example, [17], where MTSP is used for modeling the preprinted insert scheduling problem. Planning problems have also been investigated through MTSP formulations, specifically in [18, 19], where a dynamic mission planning system for multiple mobile robots operating in unstructured environments is presented (analysis of planetary exploration), or in [20], where the MTSP formulation is used to describe a path planning problem for a team of cooperative vehicles. An important and well-studied extension of the MTSP is the Vehicle Routing Problem [21, 22], where a fleet of vehicles of different capacities, based at either one or several depots, must deliver different customer demands (the number of vehicles is often considered as a minimization criterion in addition to total traveled distance).

A long-term goal of this work is to endow a team of autonomous agents (drones) with the capability of cooperative motion planning. In this application, the time available for the solution is limited and real-time algorithms providing

good approximate solutions are required. In this work, the problem of planning a set of strategies for cooperatively exploring the environment with a fleet of vehicles is modeled as a variant of the classical MTSP, referred to as the Multiagent Planning Problem (MAPP): given n nodes (targets) and m salesmen (agents) located at different depots, the MAPP seeks m tours such that each target is visited only once and by only one agent that minimizes a given cost function (specified by (8) and (9)). The paper presents a Genetic Algorithm Inspired Descent (GAID) method for obtaining good quality MAPP solutions.

The paper is organized as follows: after describing the MAPP in detail (Section 2) and an overview of how similar problems (MTSP in particular) are solved (Section 3), the proposed GA-Inspired Descent method is described in Section 4. Results are reported in Section 5, and conclusions are drawn in Section 6.

2. The Multiagent Planning Problem: Notations

Graph theory [23, 24] provides a natural framework to describe the Multiagent Planning Problem. Given $V = \{v_1, \ldots, v_m\}$, a set of m elements referred to as *vertices* (targets), and $E = \{(v_i, v_j) \mid v_i, v_j \in V\}$, a set of *edges* connecting vertices v_i and v_j, a *graph* G is defined as the pair (V, E). Here we consider only *undirected graphs*, that is, graphs whose edges are unordered pairs with the symmetry relation $(v_i, v_j) = (v_j, v_i)$. A *complete* graph is a graph where all vertices of V are connected to each other. The complete graph induced by the vertex set V is denoted by $K_m(V)$, where $m = |V|$ ($|\cdot|$ denotes the cardinality of a set) is the number of vertices. A graph $G_1 = (V_1, E_1)$ is a *subgraph* of G ($G_1 \subseteq G$) if $V_1 \subseteq V$ and $E_1 \subseteq E$ such that

$$E_1 = \left\{\left(v_i, v_j\right) \mid v_i, v_j \in V_1\right\}. \tag{1}$$

A subgraph $P = (V_1, E_1)$ is called a *path* in $G = (V, E)$ if V_1 is a set of k vertices of the original graph and

$$E_1 = \left\{\left(x_1, x_2\right), \left(x_2, x_3\right), \ldots, \left(x_{k-1}, x_k\right)\right\} \subseteq E \tag{2}$$

is the set of $k-1$ edges that connect those vertices. In other words, a path is a sequence of edges with each consecutive pair of edges having a vertex in common. Similarly, a subgraph $C = (V_2, E_2)$ of $G = (V, E)$, with

$$\begin{aligned} V_2 &= \{x_1, \ldots, x_k\} \subseteq V, \\ E_2 &= \left\{\left(x_1, x_2\right), \ldots, \left(x_{k-1}, x_k\right), \left(x_k, x_1\right)\right\} \subseteq E, \end{aligned} \tag{3}$$

is called a *cycle*. The length of a path or cycle is the number of its edges. The set of all paths and cycles of length k in G will be denoted by $\mathscr{P}_k(G)$ and $\mathscr{C}_k(G)$, respectively.

Paths and cycles with no repeated vertices are called *simple*. A simple path/cycle that includes every vertex of the graph is known as a Hamiltonian path/cycle. Graph G is called *weighted* if a *weight* (or *cost*) $w(v_i, v_j)$ is assigned to every edge (v_i, v_j). A weighted graph G is called *symmetric* if $w(v_i, v_j) = w(v_j, v_i)$. In this paper, the weight associated with each edge is the Euclidean distance between the corresponding vertices (locations); that is, $w(v_i, v_j) = w(v_j, v_i) = \|\mathbf{r}(v_i) - \mathbf{r}(v_j)\|$. The total cost $c(\cdot)$ of a path $P \in \mathscr{P}_k(G)$ is the sum of its edge weights

$$c(P) = \sum_{i=1}^{k} w\left(x_i, x_{i+1}\right). \tag{4}$$

Analogously, for a cycle $C \in \mathscr{C}_k(G)$

$$c(C) = \sum_{i=1}^{k-1} w\left(x_i, x_{i+1}\right) + w\left(x_k, x_1\right). \tag{5}$$

After having introduced the necessary notation, we are now in the position to formalize the combinatorial problems of interest. The *Subtour Problem* [25] is defined as finding a simple path $P \in \mathscr{P}_k(K_{n+1}(V))$ of length k, starting at vertex $x_1 = a$ and having the lowest cost $c(P) = \sum_{i=1}^{k} w(x_i, x_{i+1})$. If $k = n$, the problem is equivalent to finding the "cheapest" Hamiltonian path, where all the n targets in T are to be visited. The general Traveling Salesman Problem, or k-TSP, poses to find a simple cycle $C \in \mathscr{C}_{k+1}(K_{n+1}(V))$ of minimal cost starting and ending at vertex a, visiting k targets.

Let $T = \{t_1, \ldots, t_n\}$ be the set of n targets (goals) to be visited. The ith target t_i is an object located in Euclidean space whose position is specified by the vector $\mathbf{r}(t_i)$. Let $A = \{a_1, \ldots, a_m\}$ denote the set of m agents with position specified by $\mathbf{r}(a_i)$. The classical Multiple Traveling Salesman Problem can be formulated as follows. Let a denote the unique depot; that is, $a_i = a$. The *augmented vertex set* is $V = T \cup a$ and the configuration space of the problem is the complete graph $K_{n+1}(V)$. Let C_i denote a cycle of length k_i starting and ending at vertex a (the depot). The Multiple Traveling Salesmen Problem is to find m cycles (we also refer to these as *tours*) C_i of length k_i

$$\begin{aligned} \mathbb{C} &= \{C_1, \ldots, C_m\}, \\ \sum_{i=1}^{m} k_i &= n + m, \end{aligned} \tag{6}$$

such that each target is visited only once (this also implies visitation by only one agent) and the sum of the lengths (costs) of all the m tours

$$c(\mathbb{C}) = \sum_{i=1}^{m} c\left(C_i\right) \tag{7}$$

is minimal.

The Multiagent Planning Problem can be formulated similarly. For the ith agent, the augmented vertex set is given by $V_i = T \cup a_i$ and the configuration space of the problem is the complete graph $K_{n+1}(V_i)$. The weight associated with each edge is the Euclidean distance between the corresponding locations, that is, $w(v_i, v_j) = w(v_j, v_i) = \|\mathbf{r}(v_i) - \mathbf{r}(v_j)\|$, with $v_i, v_j \in V_i$, rendering $K_{n+1}(V_i)$ a weighted and symmetric graph. Let P_i denote a simple path (no repeated vertices) of length $|P_i| = k_i$ starting at vertex a_i. The optimal solution of

the Multiagent Planning Problem (MAPP) is a set $\mathscr{P}$ of m pairwise disjoint (modulo the starting points) paths P_i

$$\mathscr{P} = \{P_1, \ldots, P_m\}, \quad \sum_{i=1}^{m} k_i = n, \tag{8}$$

such that the length of the longest path

$$c_{\max}(\mathscr{P}) = \max_i c(P_i) \tag{9}$$

is minimal (a min–max problem). In other words, the m agents have to visit n targets in the least amount of time, and every target is visited only once. The number of targets visited by each agent can be different, but each agent has to visit at least one target (i.e., $k_i \geq 2$).

3. Overview of Solution Methods

The Multiple Traveling Salesmen Problem (MTSP) and the Multiagent Planning Problem are notoriously difficult to solve due to their combinatorial nature (they are NP-hard).

A common approach is to transform the MTSP into an equivalent Traveling Salesman Problem, for which solutions can be found by exact methods (e.g., branch-and-bound algorithms and linear programming [26–28]) or approximate algorithms such as Genetic Algorithms, Simulated Annealing, and Ant System [29, 30]. For example, in [31, 32] the authors proposed to transform the MTSP into an equivalent TSP by adding dummy cities and edges with ad hoc null or infinite costs. However, as stated in [33–35], transforming the MTSP into an equivalent TSP might yield an even harder problem to solve. Similar approaches are investigated in [13, 36, 37].

The first attempt to solve large-scale MTSPs is given in [33], where a branch-and-bound method (the most widely adopted technique for solving these combinatorial problems [38]) is applied to both Euclidean (up to 100 cities and 10 salesmen) and non-Euclidean problems (up to 500 cities and 10 salesmen). Branch-and-bound is also applied in [39] for solving an asymmetric MTSP up to 100 cities.

Other solution methods have also been proposed, for example, simulated annealing [18], Gravitational Emulation Local Search (GELS) algorithm [40], and evolutionary algorithms. In [41], different evolutionary algorithms, ranging from Genetic Algorithms to Particle Swarm and Monte-Carlo optimization, are compared. The MTSP with ability constraint is solved with an Ant Colony Optimization (ACO) algorithm in [34], where the MTSP is not translated into an equivalent TSP and the ACO algorithm is modified for dealing with the characteristics of the original problem. In [34] results are compared with a Modified Genetic Algorithm that solves the equivalent TSP. Linear Programming is used in [35], and similarly to [34], the original MTSP is analyzed and solved. In both [34, 35], the authors conclude that the original MTSP is easier to solve than the derived TSP. An important work is [42], where different local search heuristics are presented and compared. In [17, 29, 43, 44] Genetic Algorithms are used to minimize the sum of the salesmen path lengths together with the maximum distance traveled by each salesmen (to balance the agent workload). A Modified Genetic Algorithm on the equivalent TSP is used in [32]. Shirafkan et al. [45] proposed a hybrid method incorporating simulated annealing and Genetic Algorithm to solve the multidepot MTSP. In this MTSP variant the starting location (depot) of each agent is fixed and multiple agents can have start from the same depot. The multiobjective MTSP variant is presented in [46] and Genetic Algorithm is used to find approximate solutions. In multiobjective MTSP the total distance traveled by the agents and the balance of agent working times are the objectives. Kaliaperumal et al. [47] also used Genetic Algorithm to solve the MTSP. They developed a brand new crossover operator called modified two-part chromosome crossover and obtained better results using this new operator. A new (branch-and-cut type) exact method is presented in [48] for the heterogeneous MTSP (some targets can be visited only by a specific agent). In [49] a heuristic search method is proposed to solve the Multiagent Path Finding problem, which is similar to MAPP, except that the endpoints of the tours are also fixed. These methods work well on the problem they were constructed for, but none of them is designed to solve the MAPP. In Section 5.1.2 we benchmark our GAID against some of these methods in MTSP solving. In Section 5.2 we compare GAID against a Matlab solver which is capable of solving MAPP as well as MTSP.

4. The Multiagent Planning Problem: Approximate Solution

Given $m \geq 1$ agents and n known targets/cities to visit, the optimal team strategy (called Team Plan) is sought that allows the fleet to visit every target only once.

We represent the Team Plan as a collection of m distinct subtours. Thus, given m agents and n targets, Team Plan $\mathscr{P}$ is defined as $\mathscr{P} = \{P_1, \ldots, P_m\}$, where P_i is the path of the ith agent visiting $k_i < n$ targets. Note that this representation allows the individual subtours to have different lengths. Moreover, the Multiagent Planning Problem can also be rewritten for each ith agent to find the best possible subtour of length $k_i < n$ that satisfies the imposed cost function. We propose an optimization technique we call the Genetic Algorithm Inspired Descent (GAID) method. Briefly, a Genetic Algorithm (GA) is an optimization technique used to find approximate solutions of optimization and search problems [50–52]. Genetic Algorithms are a particular class of evolutionary methods that use techniques inspired by Darwin's theory of evolution and evolutionary biology, such as inheritance, mutation, selection, and crossover. In these systems, populations of solutions compete and only the fittest survive.

Similarly to the classical GA, GAID consists of two phases: initialization and evolution. In the initialization phase, the starting Team Plan is created (see Section 4.1), while the evolution phase (see Section 4.2) evolves Team Plan $\mathscr{P}$ toward a better quality final solution.

4.1. Initialization Phase. During the initialization phase, the starting Team Plan—a starting set of subtours—is created. Let $T_1 = \{t_1, \ldots, t_n\}$ and $A = \{a_1, \ldots, a_m\}$ be the sets of n targets and the m agents, respectively. Without loss of generality, we assume that the order of planning is $a_1, a_2, \ldots, a_m$ and that the starting subtours have similar lengths. A Team Plan is feasible if the subtours are pairwise disjoint (except possibly their starting points).

At first, a subtour P_1 (with starting point a_1) of length k_1 is chosen. Then subtour P_2 is chosen from the target set T_2 that is simply obtained by discarding from T_1 the targets visited by agent a_1. In general, the ith agent plans a subtour P_i on the targets not yet allocated previously. Obviously, this process yields a feasible Team Plan $\mathscr{P} = \{P_1, \ldots, P_m\}$.

Here we utilize three simple initialization methods.

(i) Random initialization: for each agent the targets are selected at random from the unallocated ones.

(ii) Greedy initialization: the initial Team Plan is created using a greedy approach to form feasible starting subtours. Each agent selects its targets using a Nearest Neighbor heuristics: for a given a target, the next target will be the nearest one.

(iii) TSP-based initialization: for problems where the positions of the m agents are not imposed (the m agents can start from any target $t_i \in T$), a feasible starting Team Plan can be generated by clustering the TSP solution computed on the complete graph $K_n(T)$. "Clustering" is carried out by discarding m edges from the TSP tour, in order to subdivide it "fairly", and having m starting subtours with similar costs. This initialization method introduces a degree of complexity in the overall system, since a TSP solution must be computed.

The initialization phase is an important step in the optimization process, since it directly influences the efficacy of the algorithm.

4.2. Evolution Phase. The evolution phase evolves the Team Plan, trying to design a strategy where the overall time is reduced (minimizing the cost $c_{\max}(\mathscr{P})$; see (9)).

This phase has the same mechanism of a classical Genetic Algorithm [50] with one important difference: there is only one Team Plan $\mathscr{P}$ to be evolved (i.e., the population size is 1).

At every evolution/generation step, a set of operators (see Section 4.3) is applied to the subtours $P_i \in \mathscr{P}$. If $\mathscr{P}$ improves, it is kept for the next generation step, otherwise the previous Team Plan is restored (we expect that a simulated annealing-type modification will be even more efficient). The flowchart of the GAID evolution phase is shown in Figure 1.

4.3. Team Plan Operators and Boosting. The evolution of $\mathscr{P}$ toward a near-optimal multiagent strategy is accomplished by combining the genetic materials of its subtours through the application of genetic-like operators. Three different operators have been designed: crossover, mutation, and migration. The operators are applied in a predefined order, and their application depends on a given probability, as shown in Figure 1.

The *crossover operator* (Figure 2) is applied with probability $p_{\text{crossover}}$ and combines the genetic materials of two selected subtours (called parents), replacing them with the two newly created ones (the offsprings). Parents are chosen using the best-worst selection with probability $p_{\text{best-worst}}$ (Figure 2(a)) or randomly (Figure 2(b)) with probability $1 - p_{\text{best-worst}}$. The parents are mated the classical way [50]: they are randomly halved (not necessarily at the same position) and the halves are simply swapped.

The *Mutation Operator* changes the Team Plan (with probability p_{mutation}) by randomly swapping two genes between two different subtours. The *Migration Operator* (applied with probability $p_{\text{migration}}$) removes a randomly chosen target from subtour P_i (of length k_i) and adds it to subtour P_j (of length k_j). The location at which this target is placed into subtour P_j is also chosen at random. Note that the lengths of the subtours change: k_i decreases while k_j increases.

With probability p_{boost}, each subtour is processed with a *heuristic boosting* technique. The 2-opt method [53–55] is adopted here to directly improve the quality of subtours (it replaces subtours with better ones from their "neighborhood"). The 2-opt method determines whether the inequality

$$c\left(x_i, x_{i+1}\right) + c\left(x_j, x_{j+1}\right) > c\left(x_i, x_j\right) + c\left(x_{i+1}, x_{j+1}\right) \quad (10)$$

between the four vertices x_i, x_{i+1}, x_j, and x_{j+1} of a path/subtour holds, in which case edges (x_i, x_{i+1}) and (x_j, x_{j+1}) are replaced with edges (x_i, x_j) and (x_{i+1}, x_{j+1}), respectively. This method provides a shorter path without intersecting edges.

5. Results

An extensive set of simulations were run to test the performance of the proposed GAID method. Unless otherwise specified, simulations were run for 150000 generation steps, while the crossover, mutation, migration, and boosting (2-opt) operators were applied with probabilities $p_{\text{crossover}} = 0.7$, $p_{\text{mutation}} = 0.4$, $p_{\text{migration}} = 0.6$, and $p_{\text{boost}} = 0.3$, respectively. For the application of the crossover operator, the probability of best-worst selection is $p_{\text{best-worst}} = 0.5$ (thus the probability for random selection is $1 - p_{\text{best-worst}} = 0.5$).

5.1. Comparison with Structured and Known Solutions. In this section, GAID is compared with known results. In order to make the comparison meaningful, we also imposed the same constraints (if any) used in the referenced works.

5.1.1. An Example Problem with Known Solution. In this section we present a 9-81 MAPP (9 agents, 81 targets), in which the agents and targets form 9 tightly packed clusters as shown in Figure 3. Each cluster contains an agent and 9 targets. Since the distances among the clusters are considerably longer than those among the points of a cluster, the optimal solution can be found by treating each cluster as a

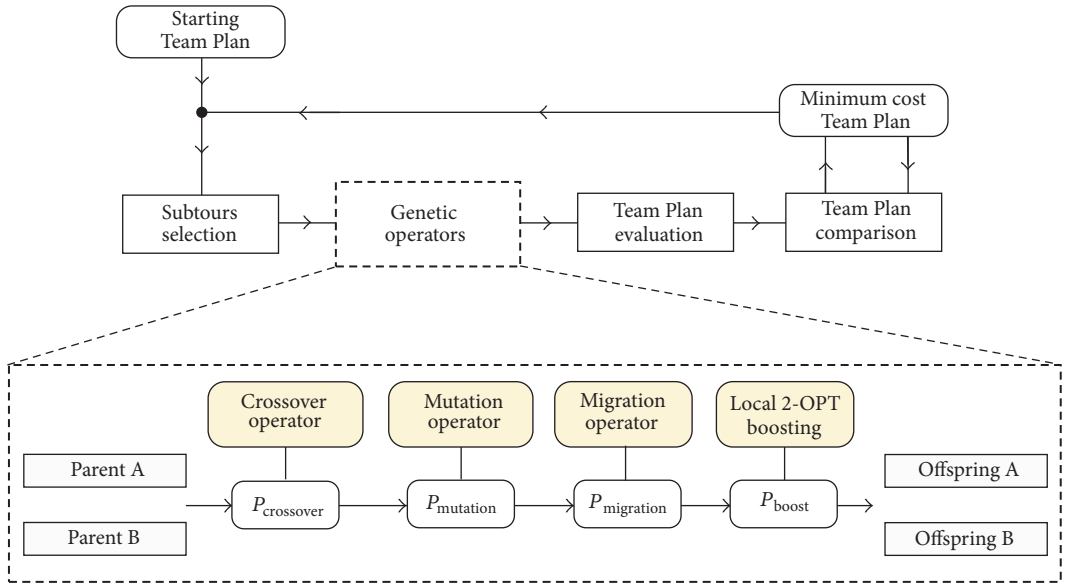
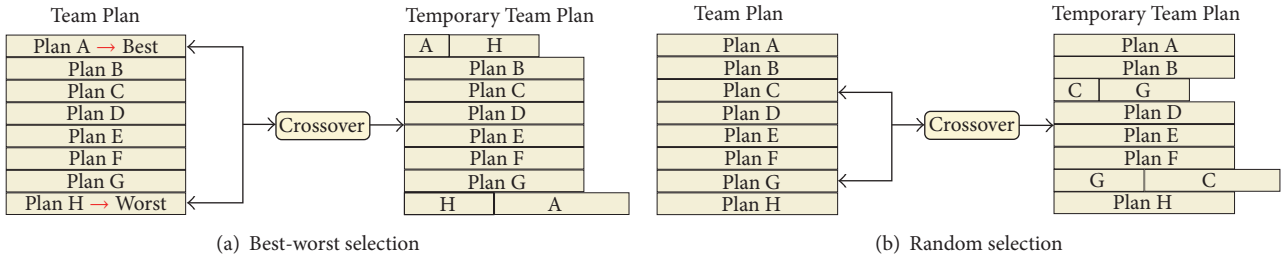


FIGURE 1: Flowchart of the evolution phase, together with the sequence of operators.



(a) Best-worst selection

(b) Random selection

FIGURE 2: Crossover operator. (a) In case of best-worst selection the best and the worst subtours are mated (with probability $p_{\text{best-worst}}$). (b) By random selection the two subtours are selected at random (with probability $1 - p_{\text{best-worst}}$).

1-9 MAPP (1 agent, 9 targets). The tour lengths are between 356.1 and 548.6, meaning that the optimal cost (the length of the longest tour) is 548.6 (see (9)). In Figure 3 we show the random and greedy initialization for this MAPP. In both cases some of the subtours contain a higher number of targets than others. We intentionally set the initialization this way to further challenge the algorithm. The initial tour lengths vary between 1944.9–15302.3 for the random, and 175.3–1627.9 for the greedy initialization. Consequently, the initial Team Plan costs are 15302.3 and 1627.9, respectively.

The convergence of the Team Plan cost for both simulations is shown in Figure 4 together with the cost of the optimal solution (548.6). At the end of the simulations the costs of the subtours varied between 378.3–612.3 and 389.9–594.1 for solutions starting from randomly initialized and greedy initialized Team Plans, respectively.

The final Team Plans of the two simulations are shown in Figure 5. We see that those long edges which initially connected targets belonging to different clusters were completely eliminated. The longest subtour connects the targets in the middle-top cluster for both simulations.

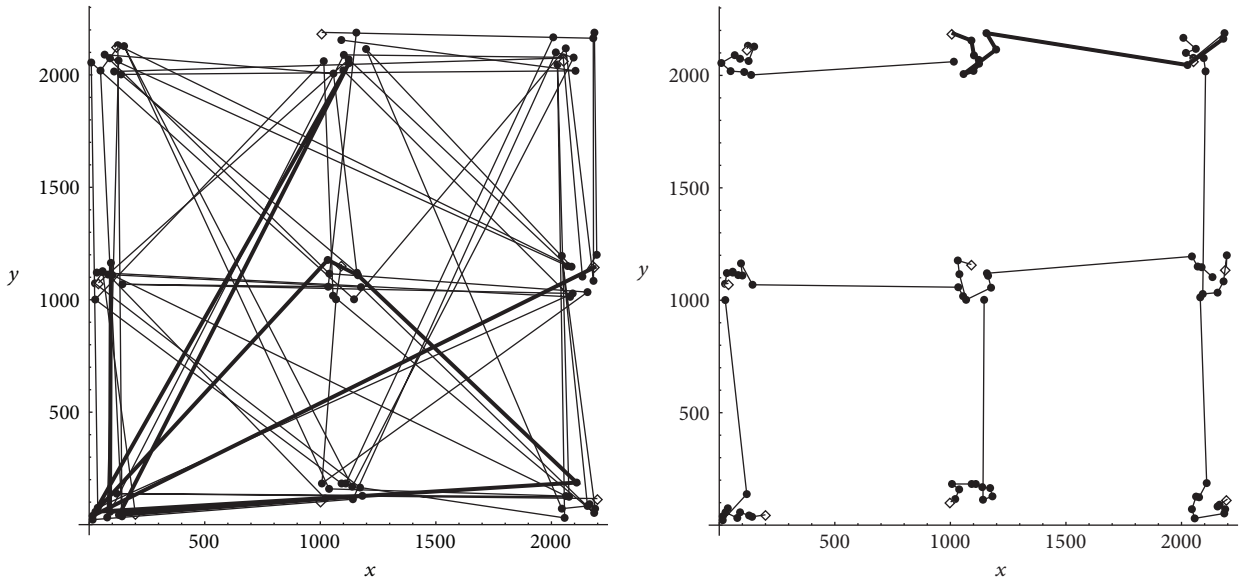
5.1.2. Comparison with Evolutionary Algorithms. We also compare our method with the results reported in [34], where an Ant Colony Optimization algorithm is compared with the Modified Genetic Algorithm (MGA) of [32]. We note here that these methods cannot solve the MAPP, therefore six MTSPs, all taken from well-known TSPLIB [56], are solved. The number of agents is set to $m = 5$, and their starting location is the first target in the corresponding problem data file. Since a single depot is considered, only the greedy initialization method is used. In both [32, 34], rounded costs are considered. Therefore, for meaningful comparison, we need to minimize the following cost function:

$$c_{\text{round}}(\mathbb{C}) = \sum_{i=1}^{m} \text{round}\left(c\left(C_i\right)\right). \tag{11}$$

In addition, the maximum length of the cycles, $k_{\max}$, is limited to 20 (note that since MTSPs are considered here, we need to modify the Team Plan using cycles instead of paths). Results based on 100 simulations are presented in Table 1. In this case, our results outperform those reported in the above

Table 1: Comparison between the proposed GAID, the Ant Colony Optimization, and the Modified Genetic Algorithm methods. Results are averaged over 100 simulations. The number of targets is included in the TSPLIB problem name. k_{max} is the maximum number of targets an agent can visit. For each case, the fixed number of $m = 5$ agents is considered, starting from the same location (one-depot problem). Only the greedy initialization method is used.

TSPLIB Problem	k_{max}	ACO		MGA		GAID		Difference of means	
		Min	Mean	Min	Mean	Min	Mean	ACO	MGA
pr76	20	178597	180690	157444	160574	152722	156503.9	−15.4%	−2.6%
pr152	40	130953	136341	127839	133337	114698	126128.8	−8%	−5.7%
pr226	50	167646	170877	166827	178501	152198	158073.9	−8%	−12.9%
pr299	70	82106	83845	82176	85796	70059	71705.1	−16.9%	−19.6%
pr439	100	161955	165035	173839	183698	136169	138655.5	−15.6%	−24.5%
pr1002	220	382198	387205	427269	459179	311492	319240.4	−17.5%	−30.5%



(a) 9-81 MAPP: initial plan with random initialization

(b) 9-81 MAPP: initial plan with greedy initialization

Figure 3: The GAID algorithm was tested on a MAPP which contains 81 targets forming 9 clusters on a 2200 × 2200 square domain. An agent was placed into each cluster (white diamonds); thus the number of agents is $m = 9$. The initial cost based on (9) is 15302.3 for random and 1627.9 for greedy initialization. The longest subtour is highlighted with thick line.

references. Figure 6 also shows the solution obtained for the pr76 TSPLIB problem.

5.1.3. Comparison with Heuristics. We compare our method against the results reported in [42], where different heuristics for solving MTSP instances are proposed and compared.

In [42], a no-depot MTSP variant is considered (the agents do not have a predefined starting location), and the min–max optimization problem is solved ((9) is minimized). In addition, the number of salesmen is fixed and no constraints on the plan lengths are imposed. Note that since we are comparing our method with MTSP results, we need to modify our Team Plan accordingly, using cycles instead of paths. In addition, since in [42] rounded distances are considered, for meaningful comparison the cost function (see (9)) is modified accordingly:

$$c_{\text{maxround}}(\mathbb{C}) = \max_i \text{round}\left(c\left(C_i\right)\right). \quad (12)$$

For each test case, our results are compared with only the best ones of [42]. We also report the name of the Heuristic with which each referenced solution has been obtained (please refer to [42] for a more detailed description of the adopted heuristic methods). Tables 2 and 3 show the results obtained by initializing the Team Plan with either the greedy or the TSP-based initialization methods, respectively. Results are averaged over 100 simulations.

In each test case, GAID returns good solutions, regardless of the applied initialization method. In general, our best solutions are close to the ones from the literature and in a few cases (e.g., in the berlin52 or the pr264 problems) are even better. The greedy initialization method seems to provide a better initial Team Plan in those problems where the distribution of targets has a well-defined structure (the pr264 problem), while in "small-size" problems (i.e., berlin52) the TSP-based method is preferable (even if the obtained improvement does not justify its required complexity, time,

Table 2: Heuristics comparison for 100 simulations. The greedy initialization method was used.

TSPLIB Problem	Number of Agents	Heuristics			Greedy initialization		Errors	
		Method	Minimum	Mean	Minimum	Mean	Minimum	Mean
berlin52	4	Bisection	2182	2204.3	2183	2422.4	0.04%	9.8%
	5	k-split	1713	1739.7	1825	2160.4	6.5%	24.2%
	6	SGH	1531	1585	1611	1905.8	—	20.2%
		Bisection	1476	1643.1			9.1%	—
kroA100	4	SGH	5955	6096.7	6000	6690.3	0.7%	9.7%
	5	k-split	4629	5025.9	4964	5620.1	7.2%	11.8%
	6	k-center	4200	4429.4	4370	5038.4	4%	—
		k-means	4230	4234.6			—	18.9%
bier127	4	Bisection	32423	32757.5	32434	35740.9	0.03%	9.1%
	6	k-split	22815	23071.7	24608	26993.3	7.8%	16.9%
pr264	4	Bisection	12705	12705	12196	13830.3	−4.1%	8.8%
	6	k-means	8526	9131.6	9000	10256.3	5.5%	—
		Bisection	8739	9051.6			—	13.3%

Table 3: Heuristics comparison for 100 simulations using the TSP-based initialization.

TSPLIB Problem	Number of Agents	Heuristics			TSP-based initialization		Errors	
		Method	Minimum	Mean	Minimum	Mean	Minimum	Mean
berlin52	4	Bisection	2182	2204.3	2088	2359.6	−4.5%	7%
	5	k-split	1713	1739.7	1804	2014.8	5.3%	15.8%
	6	SGH	1531	1585	1576	1801.2	—	13.6%
		Bisection	1476	1643.1			6.7%	—
kroA100	4	SGH	5955	6096.7	6000	6443.6	0.7%	5.7%
	5	k-split	4629	5025.9	4796	5314.6	3.6%	5.7%
	6	k-center	4200	4429.4	4310	4693.9	2.6%	—
		k-means	4230	4234.6			—	10.8%
bier127	4	Bisection	32423	32757.5	32948	34606	1.6%	5.6%
	6	k-split	22815	23071.7	24290	26497.1	6.4%	14.8%
pr264	4	Bisection	12705	12705	13400	19135.3	5.4%	50.6%
	6	k-means	8526	9131.6	8629	14395.8	1.2%	—
		Bisection	8739	9051.6			—	59%

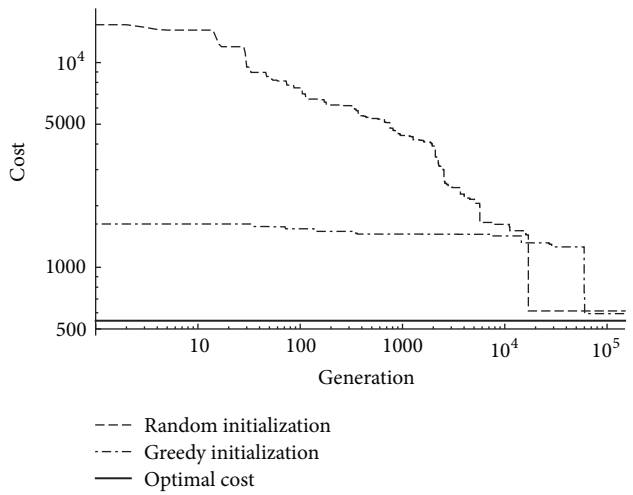


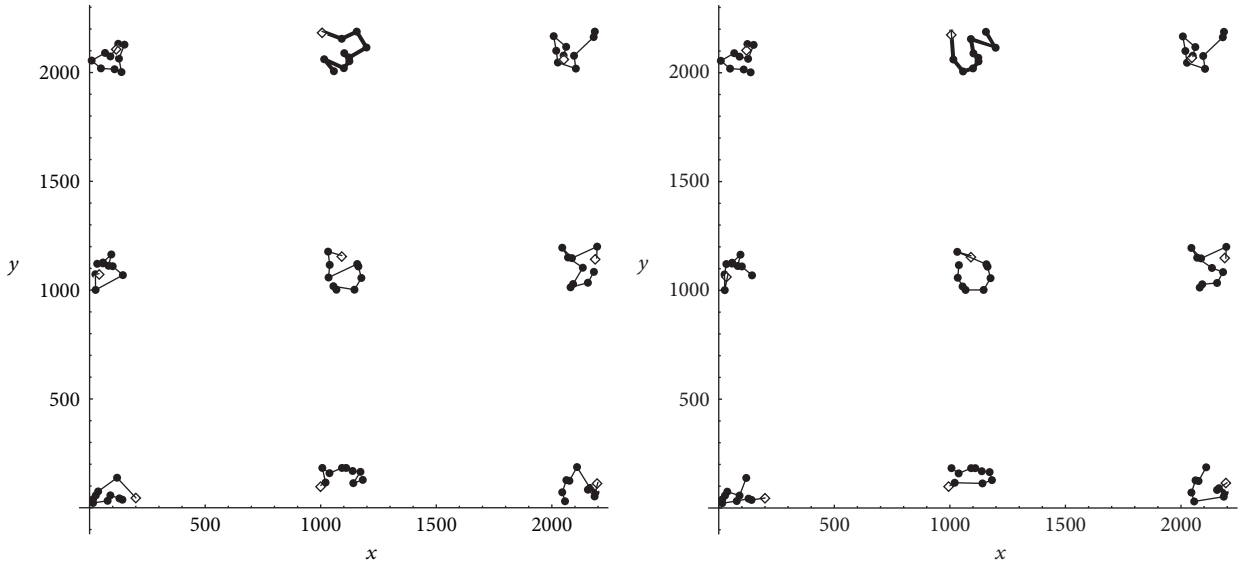
Figure 4: Convergence of the Team Plan cost for the 9-81 MAPP shown in Figure 3.

and computational costs). Even though these heuristics are superior in solving MTSP compared to GAID, they cannot solve MAPP, while our method can solve both.

Figure 7 shows the solutions obtained by applying the greedy (Figure 7(a)) and the TSP-based (Figure 7(b)) initialization methods for the kroA100 problem (with $m = 5$ agents). Compared with [42], the TSP-based method and the greedy method result in final solutions that are only 3.6% and 7.2% worse than the cited one, respectively.

5.2. Comparison with Other Software. In this section we test GAID against a freely available Matlab MTSP solver based on a Genetic Algorithm [57].

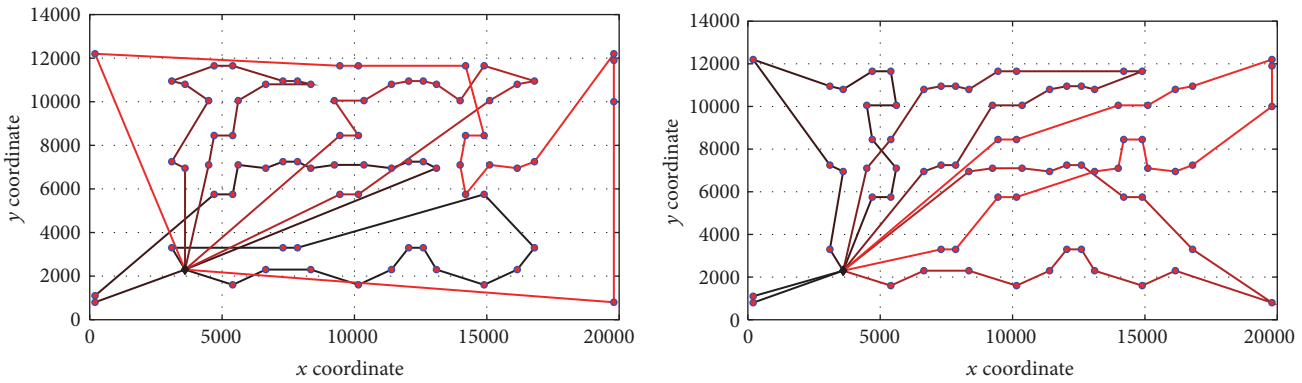
The number of agents is fixed, and the minimum path size, $k_{\min} = 2$, is imposed (this way, solutions with paths composed of only one target are avoided). The cost function, which is specified by (9), is minimized, and targets are randomly generated over the unit square map. In all simulations,



(a) 9-81 MAPP problem: final plan after random initialization

(b) 9-81 MAPP problem: final plan after greedy initialization

FIGURE 5: The GAID algorithm found 9 subtours which connect targets belonging to the same cluster for the 9-81 MAPP with both initialization strategies. The final costs are 612.3 and 594.1, respectively.



(a) pr76 TSPLIB problem: starting Team Plan

(b) pr76 TSPLIB problem: final Team Plan

FIGURE 6: The MTSP is solved for the pr76 TSPLIB target configuration. The maximum plan length is $k_{\max} = 20$ targets and the number of agents is $m = 5$. The cost function specified by (11) is minimized, and the comparison between the starting and the final Team Plans is shown. In (a) the starting Team Plan, with cost equal to 194618, is shown, while in (b) the final Team Plan, with cost equal to 152722, is reported. These solutions are 15.4% and 2.6% better than the referenced ones, obtained by implementing an Ant Colony Optimization method and a Modified Genetic Algorithm, respectively.

the greedy initialization method is adopted. Since the Matlab MTSP solver is based only on a Genetic Algorithm and no heuristics is used, we also run a set of tests with $p_{\text{boost}} = 0\%$.

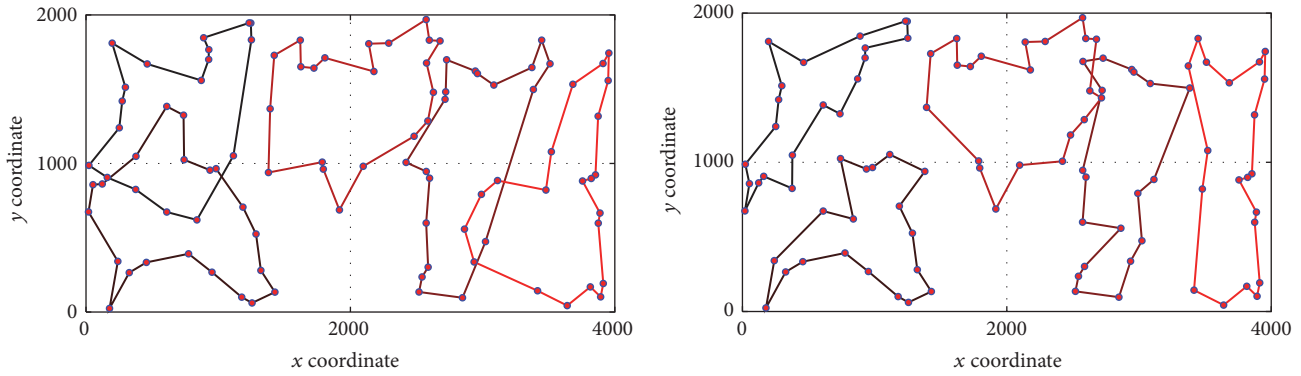
Table 4 shows the obtained results, averaged over 100 simulations. Our method clearly outperforms the Matlab solver (clearly, with the 2-opt method the results are much better). Figure 8 shows two examples where the Matlab MTSP solver solution is compared with the GAID ones.

5.3. Solution for a Large Example Problem. We tested our algorithm on a 20-400 MAPP. The initial Team Plan was constructed by means of the greedy initialization method, such that each subtour visits 10 targets. The simulation was ran for 1000000 generations.

Figures 9 and 10 summarize the results. The initial cost of 1.94 was reduced by 43.5% (to 1.08) by the end of the 150000th generation and by a further 5.7% (to 0.94) till the end of the simulation.

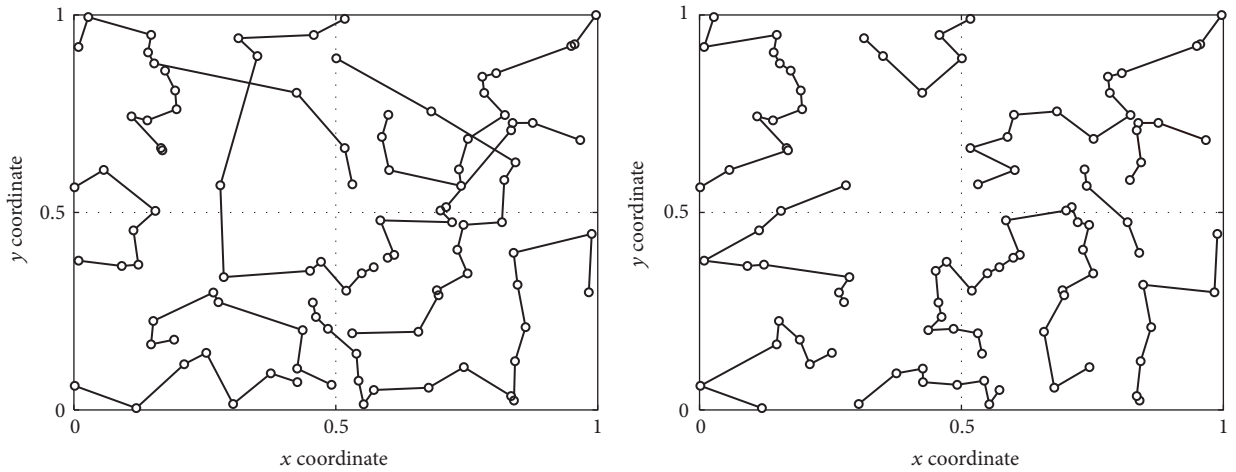
6. Conclusions and Future Work

This paper describes the Multiagent Planning Problem, a variant of the classical Multiple Traveling Salesman Problem. Our solution method uses a simplified Genetic Algorithm,



(a) kroA100 problem: greedy initialization (b) kroA100 problem: TSP-based initialization

FIGURE 7: Solving the MTSP by the greedy and the cluster initialization methods. In both cases, $m = 5$ agents are considered, and the target set is kroA100. In (a), the greedy initialization method is used and the final obtained Team Plan is shown. The cost is 4964, and it is 7.2% worse than the referenced one. In (b), the TSP-based initialization method is used and the obtained final Team Plan is shown. The cost is 4796, which is only 3.6% worse than the referenced one.



(a) Matlab solution (b) Proposed method

FIGURE 8: GAID has been compared with the Matlab MTSP solver. In this example, MAPP is solved for 100 targets randomly distributed in the unit square. $m = 10$ agents are considered. The minimum path length is $k_{\min} = 2$. The cost function (see (9)) is minimized. In (a) the solution obtained by running the MTSP Matlab solver is shown, with cost 7.92. In (b) our solution is shown with cost 6.37, which is 19.5% better than the Matlab MTSP solution.

called Genetic Algorithm Inspired Descent (GAID), capable of solving both the MTSP and the MAPP. GAID takes an initial Team Plan and applies various genetic operators to decrease the length of the longest path. We benchmarked GAID against other evolutionary algorithms and heuristics. GAID outperformed the Ant Colony Optimization and the Modified Genetic Algorithm. Even though the heuristics specifically developed for MTSP (e.g., k-split, bisection) outperformed GAID, these methods cannot solve MAPP. GAID proved to be much better than the Matlab MAPP solver. GAID is also easy to implement.

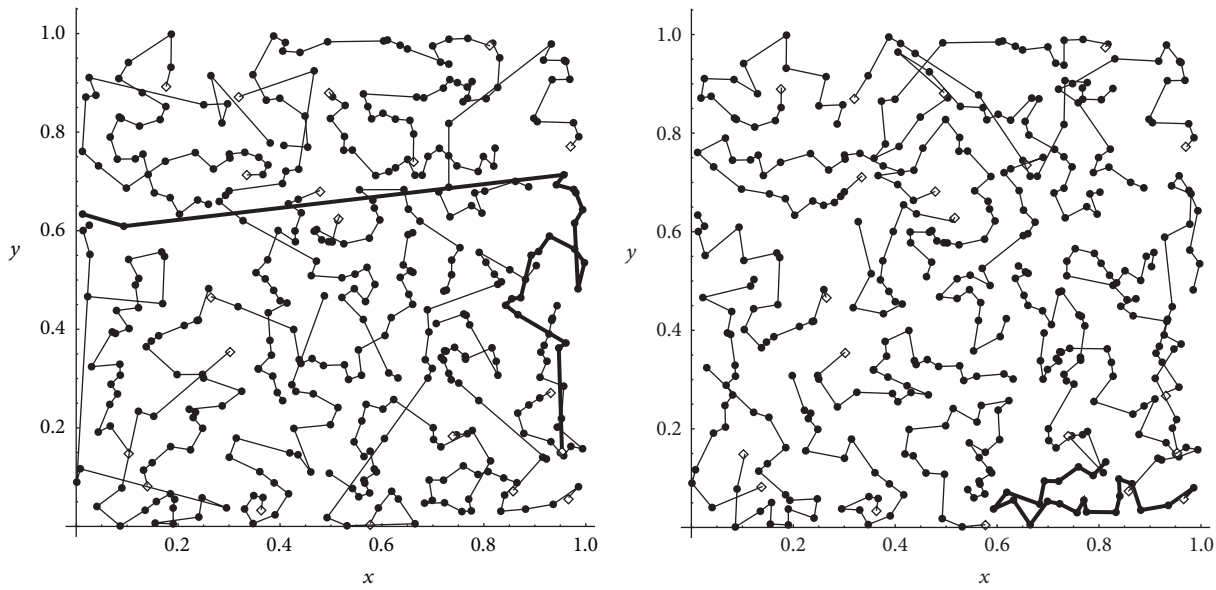
A long-term goal of this work is to endow a team of autonomous agents (drones) with the capability of cooperative motion planning. In this application, the time available for the solution is limited and fast algorithms providing good approximate solutions prevail. The results presented here show the success of the approach, demonstrating how a simple method can solve an otherwise hard combinatorial problem.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work relates to the scientific programs "Development of quality-oriented and harmonized R+D+I strategy and the functional model at BME" (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002) and "Talent care and cultivation in the scientific workshops of BME" (Project ID: TÁMOP-4.2.2/B-10/1-2010-0009).



(a) 20-400 MAPP: initial Team Plan with greedy initialization

(b) 20-400 MAPP: final Team Plan after the 1000000th generation

Figure 9: The GAID algorithm was performed on a MAPP with 20 agents and 400 targets located in the unit square. The Team Plan cost was 1.91 after the initialization and was reduced to 1.08 by the end of the 150000th generation and to 0.94 by the end of the simulation. The longest subtour is highlighted by a thick line.

Table 4: Comparison between the proposed GAID method and a Matlab MTSP code. For each test, $k_{\min} = 2$ and the greedy initialization method is used. Targets are randomly generated over the unit square.

Test case	Matlab code			Proposed method				Cost variation	
	Min	Mean	Max	p_{boost}	Min	Mean	Max	Min	Mean	Max
100 targets	7.14	8.19	9.21	30%	6.37	6.72	7.11	−10.7%	−17.9%	−22.8%
10 agents				0%	6.67	7.14	7.58	−6.5%	−12.8%	−17.7%
200 targets	15.64	17.5	19.65	30%	9.55	9.89	10.25	−39%	−43.5%	−47.8%
20 agents				0%	10.05	10.6	11.23	−35.7%	−39.4%	−42.8%

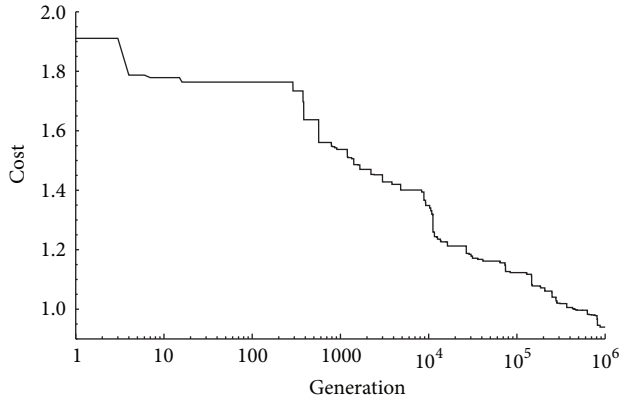


Figure 10: Convergence of the Team Plan cost for the 20-400 MAPP shown in Figure 9.

References

[1] NASA, Mars Exploration Rover missions, January 2010, http://marsrovers.nasa.gov.

[2] S. Hayati, R. Volpe, P. Backes et al., "The Rocky 7 rover: a Mars sciencecraft prototype," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '97)*, vol. 4, Albuquerque, NM, USA, April 1997.

[3] E. T. Baumgartner, "In-situ exploration of Mars using rover systems," in *Proceedings of the AIAA Space 2000 Conference*, 2000-5062, Long Beach, Calif, USA, September 2000.

[4] A. Birk and S. Carpin, "Rescue robotics—a crucial milestone on the road to autonomous systems," *Advanced Robotics*, vol. 20, no. 5, pp. 595–605, 2006.

[5] S. Sarieland and H. L. Akin, "A novel search strategy for autonomous search and rescue robots," in *Robo Cup 2004: Robot Soccer World Cup VIII*, vol. 3276, pp. 459–466, Springer, Berlin, Germany, 2005.

[6] A. Jacoff, E. Messina, and J. Evans, "Experiences in deploying test arenas for autonomous mobile robots," in *NIST Special Publication*, pp. 87–94, 2002.

[7] S. Carpin, J. Wang, M. Lewis, A. Birk, and A. Jacoff, "High fidelity tools for rescue robotics: results and perspectives," in *RoboCup 2005: Robot Soccer World Cup IX*, vol. 4020, pp. 301–311, 2006.

[8] G. Giardini and T. Kalmár-Nagy, "Centralized and distributed path planning for multi-agent exploration," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pp. 2701–2712, Hilton Head Island, SC, USA, August 2007.

[9] G. Giardini and T. Kalmár-Nagy, "Genetic algorithm for multi-agent space exploration," in *Proceedings of the AIAA infoTech at Aerospace Conference*, vol. 2, pp. 1146–1160, May 2007.

[10] S. Mitrovic-Minic and R. Krishnamutri, "The Multiple Traveling Salesman Problem with Time Windows: bounds for the minimum number of vehicles," Tech. Rep. TR 2002-11, FU CS School, 2002.

[11] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[12] S. Hong and M. W. Padberg, "A note on the symmetric multiple traveling salesman problem with fixed charges," *Operations Research*, vol. 25, no. 5, pp. 871–874, 1977.

[13] A. Singh and A. S. Baghel, "A new grouping genetic algorithm approach to the multiple traveling salesperson problem," *Soft Computing*, vol. 13, no. 1, pp. 95–101, 2009.

[14] D. S. Johnson and L. A. McGeoch, "The Traveling Salesman Problem: a case study in local optimization," in *Local Search in Combinatorial Optimization*, pp. 215–310, 1997.

[15] GaTech, Traveling Salesman Problem, January 2010, http://www.tsp.gatech.edu.

[16] G. Gutin and A. Punnen, *The Traveling Salesman Problem and Its Variations*, vol. 12 of *Combinatorial Optimizations*, Kluwer Academic, Norwell, Mass, USA, 2002.

[17] A. E. Carter and C. T. Ragsdale, "Scheduling pre-printed newspaper advertising inserts using genetic algorithms," *Omega*, vol. 30, no. 6, pp. 415–421, 2002.

[18] A. T. Stentz and B. Brummit, "GRAMMPS: a generalized mission planner for multiple mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1998.

[19] A. T. Stentz and B. Brummit, "Dynamic mission planning for multiple mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 1996.

[20] Z. Yu, L. Jinhai, G. Guochang, Z. Rubo, and Y. Haiyan, "An implementation of evolutionary computation for path planning of cooperative mobile robots," in *Proceedings of the 4th World Congress on Intelligent Control and Automation*, vol. 3, pp. 1798–1802, Shanghai, China, June 2002.

[21] UMA, The VRP Web, January 2009, http://neo.lcc.uma.es/radiaeb/WebVRP.

[22] F. B. Pereira, J. Tavares, P. Machado, and E. Costa, "GVR: a new genetic representation for the vehicle routing problem," in *Proceedings of the 13th Irish Conference on Artificial Intelligence and Cognitive Science (AICS '02)*, pp. 95–102, Limerick Ireland, September 2002.

[23] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, Macmillan, London, UK, 1976.

[24] R. Diestel, *Graph Theory*, Springer, 2005.

[25] G. Giardini and T. Kalmár-Nagy, "Genetic algorithm for combinatorial path planning: the subtour problem," *Mathematical Problems in Engineering*, vol. 2011, Article ID 483643, 31 pages, 2011.

[26] S. Tschoke, R. Luling, and B. Monien, "Solving the traveling salesman problem with a distributed branch-and-bound algorithm on a 1024 processor network," in *Proceedings of the IEEE 9th International Parallel Processing Symposium*, pp. 182–189, Santa Barbara, Calif, USA, April 1995.

[27] E. Balas and P. Toth, "Branch and bound methods for the traveling sales man problem," Tech. Rep., DTIC Document, 1985.

[28] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, New York, NY, USA, 1986.

[29] A. E. Carter, *Design and application of genetic algorithms for the multiple traveling salesperson assignment problem [Ph.D. thesis]*, Department of Management Science and Information Technology, Virginia Polytechnic Institute and State University, 2003.

[30] M. Kulich, J. Kubalik, J. Klema, and J. Faigl, "Rescue operation planning by soft computing techniques," in *Proceedings of the IEEE 4th International Conference on Intelligent Systems Design and Application*, pp. 103–109, Budapest, Hungary, 2004.

[31] S. Gorenstein, "Printing press scheduling for multi-edition periodicals," *Management Science*, vol. 16, no. 6, pp. B-373–B-383, 1970.

[32] L. Tang, J. Liu, A. Rong, and Z. Yang, "A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron and Steel Complex," *European Journal of Operational Research*, vol. 124, no. 2, pp. 267–282, 2000.

[33] B. Gavish and K. Srikanth, "An optimal solution method for large-scale multiple traveling salesmen problems," *Operations Research*, vol. 34, no. 5, pp. 698–717, 1986.

[34] "First international conference on innovative computing, information and control—TOC," in *Proceedings of the First International Conference on Innovative Computing, Information and Control—Volume I (ICICIC '06)*, vol. 3, August 2006.

[35] I. Kara and T. Bektas, "Integer linear programming formulations of multiple salesman problems and its variations," *European Journal of Operational Research*, vol. 174, no. 3, pp. 1449–1458, 2006.

[36] C. S. Orloff, "Routing a fleet of M vehicles to/from a central facility," *Networks*, vol. 4, no. 2, pp. 147–162, 1974.

[37] M. Bellmore and S. Hong, "Transformation of multisalesman problem to the standard traveling salesman problem," *Journal of the Association for Computing Machinery*, vol. 21, no. 3, pp. 500–504, 1974.

[38] G. Laporte, Y. Nobert, and S. Taillefer, "A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem," *Mathematical Modelling*, vol. 9, no. 12, pp. 857–868, 1987.

[39] A. I. Ali and J. L. Kennington, "The asymmetric M-travelling salesman problem: a duality based branch-and-bound algorithm," *Discrete Applied Mathematics*, vol. 13, no. 2-3, pp. 259–276, 1986.

[40] A. S. Rostami, F. Mohanna, H. Keshavarz, and A. A. Hosseinabadi, "Solving multiple traveling salesman problem using the gravitational emulation local search algorithm," *Applied Mathematics & Information Sciences*, vol. 9, no. 2, pp. 699–709, 2015.

[41] D. Sofge, A. Schultz, and K. De Jong, "Evolutionary computational approaches to Solving the Multiple Traveling Salesman Problem using a neighborhood attractor schema," in *Applications of Evolutionary Computing*, vol. 2279, pp. 153–162, Springer, 2002.

[42] B. Na, *Heuristics for no-depot multiple traveling salesmen problem with minmax objective [M.S. thesis]*, H. Milton School of Industrial and Engineering, Georgia Institute of Technology, Atlanta, Ga, USA, May 2006.

[43] A. E. Carter and C. T. Ragsdale, "A new approach to solving the multiple traveling salesperson problem using genetic algorithms," *European Journal of Operational Research*, vol. 175, no. 1, pp. 246–257, 2006.

[44] E. C. Brown, C. T. Ragsdale, and A. E. Carter, "A grouping genetic algorithm for the multiple traveling salesperson problem," *International Journal of Information Technology and Decision Making*, vol. 6, no. 2, pp. 333–347, 2007.

[45] M. T. Shirafkan, H. Seidgar, J. Rezaian-Zeidi, and N. Javadian, "Using a hybrid simulated annealing and genetic algorithms for non fixed destination multi-depot multiple traveling sales men problem with time window and waiting penalty," *The Journal of Mathematics and Computer Science*, vol. 4, pp. 428–435, 2012.

[46] R. I. Bolaños, M. G. Echeverry, and J. W. Escobar, "A multiobjective non-dominated sorting genetic algorithm (NSGA-II) for the multiple traveling salesman problem," *Decision Science Letters*, vol. 4, no. 4, pp. 559–568, 2015.

[47] R. Kaliaperumal, A. Ramalingam, and J. Sripriya, "A modified Two part chromosome Crossover for solving mtsp using Genetic algorithms," in *Proceedings of the International Conference on Advanced Research in Computer Science Engineering and Technology (ICARCSET '15)*, Unnao, India, March 2015.

[48] K. Sundar and S. Rathinam, "An exact algorithm for a heterogeneous, multiple depot, multiple traveling salesman problem," in *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS '15)*, pp. 366–371, Denver, Colo, USA, June 2015.

[49] G. Sharon, R. Stern, M. Goldenberg, and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 195, pp. 470–495, 2013.

[50] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional, Boston, Mass, USA, 1989.

[51] K. Bryant, *Genetic algorithms and the traveling salesman problem [Ph.D. thesis]*, Department of Mathematics, Harvey Mudd College, Claremont, Calif, USA, 2000.

[52] K. Katayama, H. Sakamoto, and H. Narihisa, "The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem," *Mathematical and Computer Modelling*, vol. 31, no. 10–12, pp. 197–203, 2000.

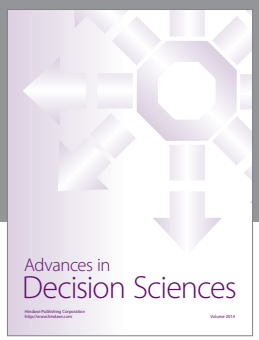
[53] M. Matayoshi, M. Nakamura, and H. Miyagi, "A genetic algorithm with the improved 2-opt method," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '04)*, vol. 4, pp. 3652–3658, IEEE, The Hague, The Netherlands, October 2004.

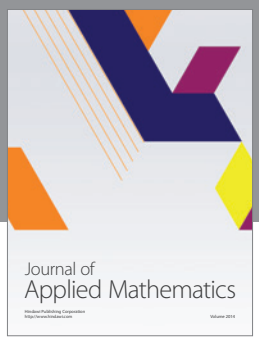
[54] J. L. Bentley, "Experiments on traveling salesman heuristics," in *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 91–99, Society for Industrial and Applied Mathematics, San Francisco, Calif, USA, January 1990.

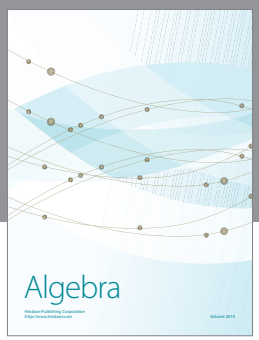
[55] H. Sengoku and I. Yoshihara, "A fast TSP solver using GA on JAVA," in *Proceedings of the 3rd International Symposium on Artificial Life, and Robotics (AROB III '98)*, pp. 283–288, 1998.

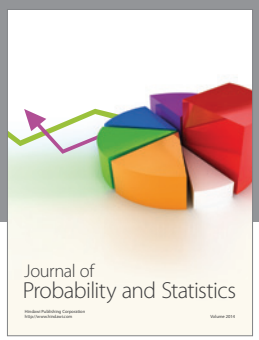
[56] G. Reinelt, "TSPLIB: a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

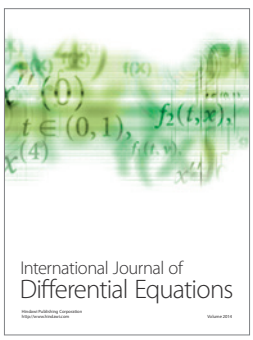
[57] Multiple Traveling Salesmen Problem—Genetic Algorithm, March 2008, http://www.mathworks.com/matlabcentral/fileexchange/.

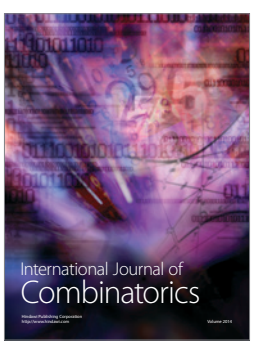


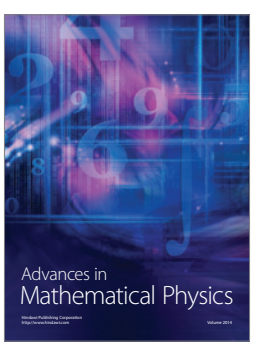


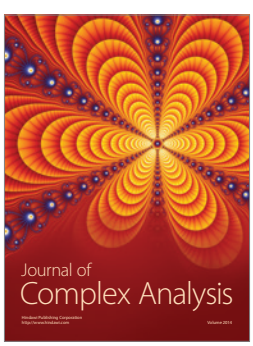


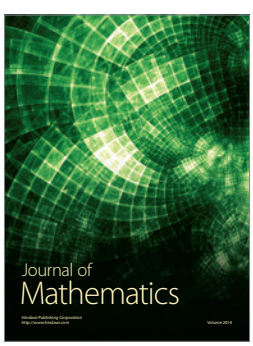


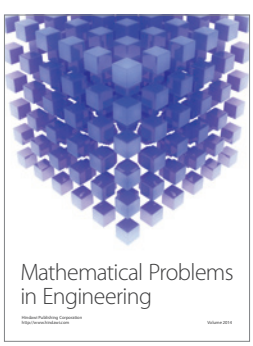


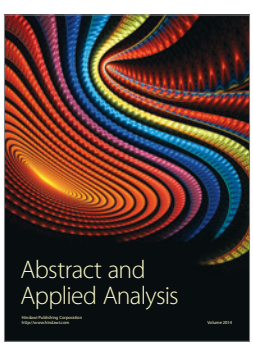


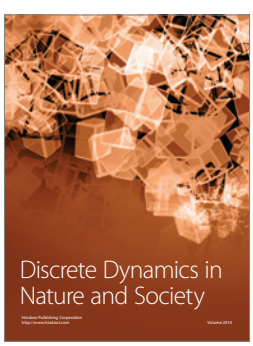


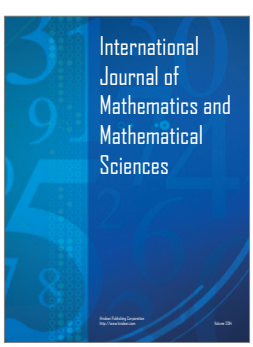


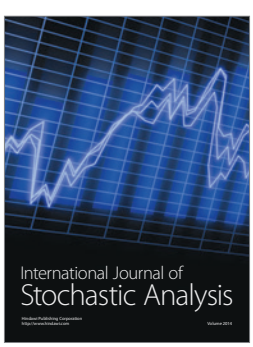












Hindawi

Submit your manuscripts at
https://www.hindawi.com