
Peer reviewed version

Link to published version (if available):
10.1145/3319619.332199

Link to publication record in Explore Bristol Research
PDF-document

This is the author accepted manuscript (AAM). The final published version (version of record) is available online via ACM at https://dl.acm.org/citation.cfm?id=3321993. Please refer to any applicable terms of use of the publisher.

University of Bristol - Explore Bristol Research
General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available: http://www.bristol.ac.uk/pure/about/ebr-terms
Decentralised Multi-Demic Evolutionary Approach to the Dynamic Multi-Agent Travelling Salesman Problem

Thomas Kent
University of Bristol
United Kingdom
thomas.kent@bristol.ac.uk

Arthur Richards
University of Bristol
United Kingdom
arthur.richards@bristol.ac.uk

ABSTRACT
This paper looks to use both centralised and decentralised implementations of Evolutionary Algorithms to solve a dynamic variant of the Multi-Agent Travelling Salesman Problem. The problem is allocating an active set of tasks to a set of agents whilst simultaneously planning the route for each agent. The allocation and routing are closely coupled parts of the same problem, this paper attempts to align the real world implementation demands of a decentralised solution by using multiple populations with well defined interactions to exploit the problem structure.

KEYWORDS
Multi Agent Travelling Salesman; Evolutionary Algorithms; Allocation and Routing; Distributed problem solving; Decision Making

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

GECCO ’19 Companion, July 13–17, 2019, Prague, Czech Republic © 2019 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-6748-6/19/07... $15.00
https://doi.org/10.1145/3319619.3321993

1 INTRODUCTION AND BACKGROUND
Reconnaissance and surveillance, search and rescue and package delivery are real-world problems relying on decision making and coordination of multiple agents [1]. Many can be modelled as being given a number of tasks to be completed and a number of agents to complete them, what is the best way to allocate tasks to agents and subsequently navigate between those tasks. The optimisation problem can be defined as a slight variation on the Multi-Agent Travelling Salesman Problem (MATSP), in both allocating a set of tasks to a number of agents and simultaneously planning the route for each agent [2] in order to minimise some given cost function. The driving question of this work is: can the real-world coordination of multiple agents [1] be solved to benefit both the solver and execution?

This work uses the flow-based formulation [2] of the Multi-Agent Travelling Salesman Problem. Let \( i \) and \( j \) denote a task from the set \( T = \{1, ..., N\} \) of tasks, the set \( A = \{1, ..., M\} \) of agents and the matrix \( c_{ij} \) to denote the cost of agent \( a \) travelling from task \( i \) to \( j \). Additionally define the binary decision variable \( x_{ij} \), which equals 1 if agent \( a \) visits task \( j \) immediately after task \( i \), and zero otherwise. The objective is to minimize the total cost of all the agents travelling between the assigned tasks that is: \[
\min_{x_{ij}} \sum_{i \in T} \sum_{j \in T} \sum_{a \in A} c_{ij}x_{ij}.
\]
This is subject to a number of standard constraints ensuring agents are used only once and tasks are all visited exactly once. Additionally, subtour elimination constraints from Bektas [2]. This paper relaxes the need for agents to start or finish at a depot by representing the agents’ current locations as dummy tasks with zero return cost, acting as their own personal depot.

2 EVOLUTIONARY ALGORITHM FOR MATSP
A MATSP suitable chromosome representation is implemented based on Tan et al. [6]. Explicitly, define \( T \) to be the set of all \( N \) tasks \( t_i \) for \( i \in \{1, ..., N\} \) and \( A \) to be the set of all agents \( a \in \{1, ..., M\} \). Then let \( \tau_k \subseteq T \) be an ordered subset, for each agent \( k \in A \), a chromosome \( X \), and solution to the MATSP is defined as \( X := \{\tau_1, ..., \tau_A\} \) such that \( \tau_a \cap \tau_b = \emptyset, \forall a \neq b \in A \). A population, \( P \), is then a set of current chromosomes, \( X_l \), defined as \( P := \{X_l\} \), for all \( l \in \{1, ..., \mu\} \), where \( \mu \) denotes population size. The fitness quality of each chromosome then corresponds to the MATSP objective, thus we seek the individual that minimises this.

The EA follows the standard three stage approach of initialisation, reproduction, selection. Three MATSP specific reproduction operators are implemented for Mutation, Crossover and Improvement. Two Mutation operators, from Qi et al. [5], swap-mutation and move-mutation, two crossover operators, Sequence-Based Crossover and Route-Based Crossover based on Potvin and Bengio [4]. In addition, an improvement heuristic operator based on the 2-opt method [3] is implemented. Finally, the selection operators, random selection...
We apply the single-population EA, cMDEA and the dMDEA to solve a set of 50 sample problems, taking place in a 200 by 200 metre area with agents’ initial locations and all tasks being randomly

placed. The methods have been implemented in Python 3.5 and run on laptop with a 2.7Ghz core i7 CPU and 16GB of RAM. For the EA a single population of $\mu = 50$ was used producing $\lambda = 25$ offspring per generation and for the multi-demic cases each deme had a population $\mu = 20$, each producing $\lambda = 10$ offspring with 5 generations produced per time-step. The trials have been run for 5 agents with 35 initial tasks, with a further 18 added periodically (once per 5 time steps). For the dMDEA we look at a range of communication radii, $r$ from 25 to 200, where importantly the agents only evolve demes corresponding to agents within $r + 10$ metres.

The objective function, total distance travelled, shown in Figure 2 clearly shows that as the communications restriction is gradually lifted the total distances of the dMDEA results tend to the cMDEA, notably, any communication radius of 125 or greater either matches or outperforms the EA. In addition, as communication range is increased the agents spend more time evolving the demes corresponding to nearby agents and thus the linear runtime increases. Clearly Figure 2 shows the relationship between the communication radius and the number of other agents to consider and the resulting run-time. Therefore there is no clear trade-off decision between ability to communicate, and thus agents you should consider, and run-time. However, as the calculations could potentially be done on board each agent and thus be done in parallel, this would result in wall-time scaling with problem size significantly closer to $O(\mu)$ than the $O(\mu^2)$ of the cMDEA.

ACKNOWLEDGMENTS

This work was funded and delivered in partnership between the Thales Group and the University of Bristol, and with the support of the UK EPSRC Grant Award EP/R004757/1 (T-B PHASE).

REFERENCES


