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

ACM Reference Format:
Thomas Kent and Arthur Richards. 2019. Decentralised Multi-Demic Evolutionary Approach to the Dynamic Multi-Agent Travelling Salesman Problem. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. 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 constraints of the problem, such as limitations to communication, need for robustness and spatial separation of agents, inform the structuring of the optimisation technique in such a way that is mutually beneficial to both the solver and execution?

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3321993

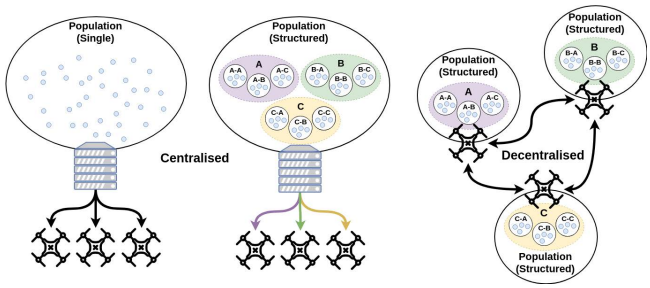

Figure 1: Population structures and flow

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_{ija} to denote the cost of agent a travelling from task i to j. Additionally define the binary decision variable x_{ija} 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: $\min_{x_{ija}} \sum_{i \in T} \sum_{j \in T} \sum_{a \in A} c_{ija} x_{ija}$. 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 μ 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*

for reproduction and *tournament selection* for determining the next generation are used.

During simulation the state of the world changes and the MATSP problem is updated. This is captured via an *update stage* to move the simulation forward by a time-step dt. It performs the following: 1) Move the Agents towards the next task in their current route; 2) Check if tasks are complete; 3) Add new tasks to the simulation (added to the end of the route of the closest agent); 4) Update the distance matrix c_{ija}. Thus the simulation begins with *initialisation*, then for every time-step the *reproduction*, *selection*, then *update* stages are run, with these repeating until all tasks are completed.

3 MULTI-DEMIC EA FOR MATSP

This paper looks to use the population-distribution island-model [6], where the global population is divided into a number of demes (distinct populations) and referred to as the Multi-Demic Evolutionary Algorithm (MDEA). Communications between these demes allow for individuals to migrate between them at pre-defined intervals. These demes are structured to align with real world execution of a MATSP where tasks are *distributed* amongst multiple agents and are completed independently.

Each agent, $k \in \{1, .., A\}$, holds a set of demes, $\mathcal{P}_k := \{P_{kl}\}$, for all $l \in \{1, .., A\}$, one deme for each other agent l as depicted in Figure 1. For each of these demes the EA is restricted to alter only parts of the solution which affect the allocation and route of agent k or agent l (the deme's agent pair). Importantly each agent has a pairwise way of 'reasoning' about potential interactions with the other agent, with tasks being exchanged only between agents who own them and/or agree to exchange, but can also use its 'personal' deme P_{kk} to improve its own route in isolation.

The MDEA uses the initialisation, reproduction, selection and update, but applied independently to each deme. After a number of generations the *exchange stage* synchronously migrates individuals between demes and generates the best current route. For all feasible exchanges between agents capable of communicating this stage carries out the following: 1) 1st knowledge update to propagate each agent's current allocation and fix/prune invalid individuals; 2) Migrate individuals whose allocation is valid for that deme's agent pair between corresponding demes of the other agents i.e $P_{kl} \leftrightarrow P_{lk}$; 3) Exchange allocations, randomly from all feasible exchanges; 4) Update each agent k's current best routes from the best individual from its personal deme P_{kk}; 5) 2nd knowledge update to propagate each agent's current allocation and fix/prune invalid individuals;

For the Centralised MDEA (cMDEA) the best solution is the best individual from *all* demes. In the Decentralised MDEA (dMDEA), restrictions on communications are implemented, meaning that agents are only able to exchange when they are within a certain proximity, r, of one another. Structuring the demes and exchanges in such a way allows exchange in a pairwise manner and in turn allows each agent to always be sure that their allocation and route does not conflict with any one else's, they are free to evolve their demes and 'reason' about potential exchanges with other agents.

4 RESULTS AND DISCUSSION

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.

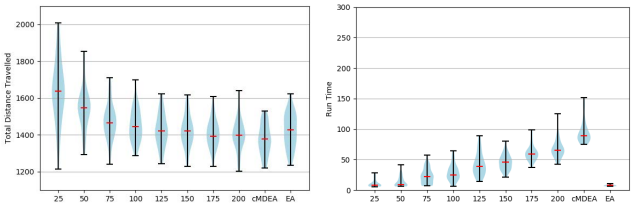

Figure 2: Communication distances: 5 Agents 35 Tasks

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 tends 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 thus the number of other agents to consider and the resulting run-time. Therefore there is a 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 $O(A)$ than the $O(A^2)$ 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

[1] Mehdi Alighanbari. 2004. *Task assignment algorithms for teams of UAVs in dynamic environments.* Ph.D. Dissertation. Massachusetts Instittue of Technology.
[2] Tolga Bektas. 2006. The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega* 34, 3 (2006), 209–219. https://doi.org/10.1016/j.omega.2004.10.004
[3] G. A. Croes. 1958. A Method for Solving Traveling-Salesman Problems. *Operations Research* 6, 6 (1958), 791–812. https://doi.org/10.1287/opre.6.6.791
[4] J. Potvin and S. Bengio. 1996. The Vehicle Routing Problem with Time Windows-Part II: Genetic Search. *INFORMS journal on Computing* 8, 2 (1996), 1–21.
[5] Y. Qi, Z. Hou, H. Li, J. Huang, and X. Li. 2015. A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows. *Computers and Operations Research* 62 (2015), 61–77. https://doi.org/10.1016/j.cor.2015.04.009
[6] K. C. Tan, Y. H. Chew, and L. H. Lee. 2006. A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Computational Optimization and Applications* 34, 1 (2006), 115–151. https://doi.org/10.1007/s10589-005-3070-3