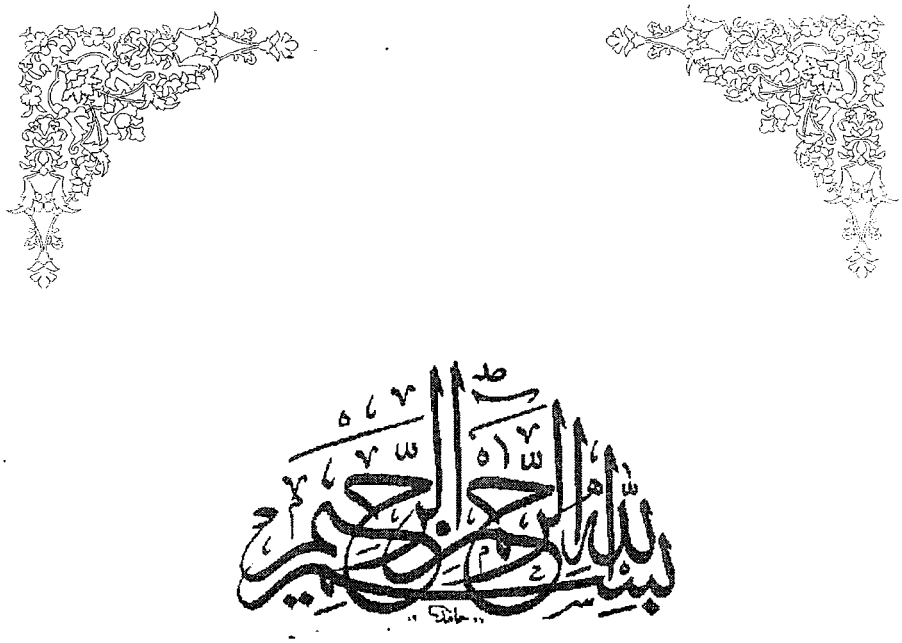
بسم الله الرحمن الرحيم

بسمه تعالی

گزارش نهایی طرح پژوهشی تحت عنوان:

حل مساله p—میانه روی شبکه‌های با وزن مثبت و منفی بوسیله الگوریتم ژنتیک

مجری: جعفر فتحعلی
عضو هیئت علمی دانشکده ریاضی

این طرح با استفاده از اعتبارات پژوهشی دانشگاه صنعتی شاهرود انجام شده است و تاریخ تصویب و خاتمه آن به ترتیب ۱۳۸۵/۳/۲۷ و ۱۳۸۵/۱۲/۲۷ می باشد.

حل مساله p–میانه روی شبکه‌های با وزن مثبت و منفی بوسیله الگوریتم ژنتیک

چکیده

فرض کنید شبکه وزندار $N = (V, E)$ شامل n راس داده شده است. در مسأله p–میانه هدف پیدا کردن یک مجموعه مانند X شامل p راس روی شبکه به گونه‌ای است که مجموع وزنی فواصل رئوس از X کمترین مقدار شود. در مسأله p–میانه کلاسیک وزن رئوس مثبت فرض میشود. در این طرح ما به حالتی توجه می کنیم که وزن رئوس می توانند منفی نیز باشند. در این حالت، که اخیراً مورد توجه قرار گرفته‌است، دو نوع تابع هدف وجود دارد. ما یک الگوریتم ژنتیک برای حل این مساله ارائه کرده و نتایج را با روش جستجوی همسایگی متغیر مقایسه می کنیم.

فهرست مندرجات

۱ مقدمه

مسأله p–میانه یکی ازمسائل اساسی در نظریه مکانیابی است و توجه بسیاری را در دهه های اخیر به خود جلب کرده است. در مسأله p–میانه روی شبکه هدف پیدا کردن مجموعه $X = \{x_۱, x_۲, \cdots, x_p\}$ شامل مکان p سرویس دهنده روی شبکه $N = (V, E)$ است به گونه‌ای که مجموع فاصله‌ها از این مجموعه تا تمام رئوس روی N کمترین مقدار شود. فاصله هر راس $v \in V$ از مجموعه X به صورت فاصله v تا نزدیکترین نقطه در X تعریف می شود. معمولا به هر راس v_i یک وزن w_i نیز تخصیص می‌یابد. که در این حالت هدف کمینه کردن مجموع فاصله ها خواهد بود. یعنی:

$$F(X) = \sum_{v_i \in N} w_i d(X, v_i)$$

که در آن $d(X,p) = \min_{x_i \in X} d(x_i, p)$ و $d(x,y)$ فاصله بین دو نقطهٔ x و y روی شبکه می‌باشد.

روشهای مختلفی که برای حل مساله p–میانه بکار برده می‌شود را می توان به چهار دسته تقسیم کرد: روش شمارشی، روشهای مبتنی بر برنامه ریزی ریاضی، روشهایی با استفاده از گرافها و روشهای ابتکاری.

اساس کار روشهای مبتنی بر برنامه ریزی ریاضی برای مسأله p–میانه استفاده از مدل برنامه ریزی خطی صفر و یک است. اولین مدل برنامه ریزی خطی صفر و یک برای این مساله توسط ریول و اسوین[۱] [۲۶] ارائه شد.

یک مدل برنامه ریزی خطی صفر و یک برای مسأله p–میانه می تواند به صورت زیر نوشته شود: فرض کنید به ازای $i = ۱, \cdots, n$ و $j = ۱, \cdots, n$ متغیرهای x_{ij} و y_j به صورت زیر تعریف شوند.

$$x_{ij} = \begin{cases} ۱ & \text{اگر رأس } i \text{ به سرویس دهنده روی راس } j \text{ اختصاص داده شود} \\ ۰ & \text{در غیر اینصورت} \end{cases}$$

$$y_j = \begin{cases} ۱ & \text{اگر رأس } j \text{ به عنوان مکان سرویس دهنده انتخاب شود} \\ ۰ & \text{در غیر اینصورت} \end{cases}$$

حال چون p سرویس دهنده مورد نیاز است پس باید محدودیت $\sum_{j=۱}^{n} y_j = p$ را داشته باشیم. همچنین هر رأس باید تنها به یک سرویس دهنده اختصاص یابد بنابراین به ازای

---

[۱] Swain

$i = ۱,\cdots,n$ داریم $\sum_{j=۱}^{n} x_{ij} = ۱$ و رأسها را تنها می‌توان به رئوسی اختصاص داد که به عنوان مکان یک سرویس دهنده درنظر گرفته شده باشند (یعنی به ازای آن رأس داشته باشیم $y_j = ۱$) بنابراین به ازای $i = ۱,\cdots,n$ و $j = ۱,\cdots,n$ داریم $x_{ij} \leq y_j$. یعنی مسأله به صورت زیر خواهد بود:

$$
\begin{array}{lll}
\min & z = \sum_{j=۱}^{n} \sum_{i=۱}^{n} w_i d_{ij} x_{ij} & \\
s.t. & & \\
 & \sum_{j=۱}^{n} x_{ij} = ۱ & i = ۱,\cdots,n \\
 & -x_{ij} + y_j \geq ۰ & i = ۱,\cdots,n \quad j = ۱,\cdots,n \\
 & \sum_{j=۱}^{n} y_j = p & \\
 & y_j \in \{۰,۱\} & j = ۱,\cdots,n \\
 & x_{ij} \in \{۰,۱\} & i = ۱,\cdots,n \quad j = ۱,\cdots,n
\end{array}
$$

روشهای ابتکاری، روشهایی مبتنی بر آزمون و خطا هستند و تضمین نمی کنند که جواب بدست آمده یک جواب بهینه باشد. این روشها زمانی به کار میروند که یک جواب نزدیک به بهینه ولی نه لزوما بهینه مورد نیاز است. یکی از اولین روشهای ابتکاری برای مسأله p—میانه الگوریتمی مبتنی بر روش آزمند[۲] است که توسط کوهن[۳] و هامبورگر[۴] [۲۱] ارائه شده است. روش آزمند یک جواب شدنی پیدا می کند که لزوما جواب نزدیک به بهینه یا بهینه موضعی نیست. بنابراین روشهای دیگری برای بهبود این جواب ارائه شدند. این روشها به عنوان روشهای بهبود بخشنده[۵] یا جستجوی ابتکاری[۶] شناخته می‌شوند. جدول ۱—۳ شامل بعضی روشهای آزمند و بهبود بخشنده است که برای مساله p—میانه ارائه شده است.

نوع دیگری از روشهای ابتکاری که برای مسأله p—میانه پیشنهاد شده است روشهای مبتنی بر دوگان مدل برنامه ریزی خطی و استفاده از روش ابتکاری کاهشی دوگان[۷] DUALOC، مربوط به ارلنکاتر[۸] [۱۳] است. چنین روشهایی توسط گالوا[۹] [۱۶]، کپتیو[۱۰]

---

[۲] Greedy
[۳] Kuhen
[۴] Hamburger
[۵] Improvement
[۶] Search Heuristic
[۷] Dual ascent heuristic
[۸] Erlenkotter
[۹] Galvao
[۱۰] Captivo

جدول ۱: بعضی روشهای آزمند و بهبود بخشنده

Heuristic method	Citation
ant colony	Levanova and Loresh (2004)
genetic algorithm	Hosage and Goodchild (1986), Dibble and Densham (1993), Alp et al. (2003), Correa et al. (2004)
greedy	Kuehn and Hamburger (1963), Cherian and Ravi (1998) Resende and Werneck (2003)
neighborhood search or *node partitioning*	Maranzana (1964)
node substitution or *interchange*	Teitz and Bart (1968), Whitaker (1983)
scatter search	Garcia-Lopez et al. (2003)
simulated annealing	Chyiyoshi and Galvao (2000), Levanova and Loresh (2004)
tabu search	Moreno-Perez et al. (1994), Voss (1996), Rolland et al (1996), Taillard (2003)
variable neighborhood search	Hansan and Mladenóvic (1997), Garcia-Lopez et al. (2002), Crainic et al. (2004)

[۱۱] و سنه[۱۱] و لورنا[۱۲] [۲۸] ارائه شده است.

روشهای کاهشی لاگرانژ[۱۳] نیز بر روی مساله اولیه اعمال شده و هم به عنوان قسمتی از روشهای دقیق و هم به عنوان روشهای ابتکاری برای حل مساله به کار گرفته شده‌اند (به [۳] و مراجع داخل آن مراجعه کنید).

روشهای مبتنی بر نظریه گراف برای حالتهای مختلفی استفاده شده است. برای مسأله p–میانه روی درخت کریو[۱۴] و حکیمی[۲۰] الگوریتمی دقیق با پیچیدگی زمانی $o(p^2n^2)$ ارائه کرده‌اند که آنرا تمیر[۱۵] [۲۹] به الگوریتمی با پیچیدگی زمانی $o(pn^2)$ بهبود بخشیده است.

[۱۱] Senne

[۱۲] Lorena

[۱۳] Lagrangean Relaxation

[۱۴] Kariv

[۱۵] Tamir

۲ مسأله p—میانه با وزن مثبت و منفی

حالتی را درنظر بگیرید که ایجاد وسیله جدید روی شبکه برای بعضی مشتریها ناخوشایند است و دور بودن وسیله جدید از بعضی نقاط و نزدیکی به بعضی نقاط دیگر مورد نظر است. این نوع مسائل مکانیابی اغلب مسائل مکانیابی نیمه—ناخوشایند[۱۶] (یا مسائل مکانیابی ناخوشایند[۱۷]،برای حالتی که همه رئوس وزن منفی دارند) نامیده می‌شوند. برای مطالعه مطالب بیشتر در مورد مسائل (نیمه-)ناخوشایند به [۸]، [۱۲]، [۲۳] و [۹] مراجعه کنید.

یک راه برای مدل کردن مسائل (نیمه-)ناخوشایند این است که به نقاطی که باید از وسایل جدید دور باشند وزن منفی و به بقیه وزن مثبت داد. از کاربردهای این مسأله می‌توان به تعیین مکان نیروگاههای هسته‌ای، انبارهای نظامی و انبارهای زباله اشاره کرد.

در حالت شبکه، مساله p—میانه با وزن مثبت و منفی اولین بار توسط بورکارد و کراروپ [۵] مورد بررسی قرار گرفت. آنها مسأله ۱—میانه با وزن مثبت و منفی روی شبکه را معرفی کرده و نشان دادند که این مسأله می‌تواند بطور خطی روی کاکتوس[۱۸] حل شود.

همچنین برای شبکه‌های با وزن مثبت و منفی بورکارد و همکاران [٦] دو نوع تابع هدف تعریف کرده‌اند: در مدل P_1 هدف کمینه کردن مجموع کمینه وزنی فاصله‌ها روی همه زیر مجموعه های $X \in N$ که $|X| = p$ است. یعنی

$$P_1 : \qquad f_1(X) = \sum_{i=1}^{n} \min_{1 \le j \le p} (w_i d(x_j, v_i)) \qquad (1)$$

در مدل P_2 هدف کمینه کردن مجموع وزنی کمینه فاصله‌ها روی همه زیر مجموعه های $X \in N$ که $|X| = p$ است. یعنی

$$P_2 : \qquad f_2(X) = \sum_{i=1}^{n} w_i \min_{1 \le j \le p} d(x_j, v_i) \qquad (2)$$

توجه داریم که اگر تمام وزنها مثبت باشند آنگاه $f_1(X) = f_2(X)$ و هر دو مسأله تبدیل به مسأله p—میانه کلاسیک می‌شوند.

برای مسأله P_1 به ازای $p = 2$ بورکارد و همکاران [٦] الگوریتمهایی با پیچیدگی

---

[۱۶] Semi-obnoxious

[۱۷] Obnoxious

[۱۸] کاکتوس گرافی است که در آن هر دو دور حداکثر دارای یک رأس مشترک باشند.

۲.۳ کدگذاری

در کدگذاریی که ما برای مساله p--میانه با استفاده از الگوریتم ژنتیک انجام می‌دهیم کروموزمها دقیقاً p ژن دارند که هر ژن متناظر است با اندیس راسی که به عنوان سرویس‌دهنده انتخاب می شود. به عنوان مثال {۷، ۵، ۲، ۱۰، ۱۴} یک کروموزم متناظر با یک جواب برای مساله‌ای با پنج میانه است که رئوس ۷، ۵، ۲، ۱۰ و ۱۴ به عنوان مکان سرویس دهنده‌ها انتخاب شده‌اند.

۳.۳ تعیین مطلوبیت

مطلوبیت یک کروموزم مقدار تابع هدف مساله به ازای جواب متناظر با این کروموزم می باشد. برای تعیین مقدار تابع هدف مسائل P_1 و P_2 باید به نحوه تخصیص نقاط تقاضای با وزن منفی به سرویس دهنده ها در هر مساله توجه کرد. در مساله P_1 نقاط با وزن نامنفی به نزدیکترین سرویس دهنده و نقاط با وزن منفی به دورترین سرویس دهنده تخصیص می یابند در حالیکه در مساله P_2 هر دو نوع نقاط با وزن نامنفی و منفی به نزدیکترین سرویس دهنده تخصیص می یابند.

۴.۳ جمعیت

اندازه جمعیت و جمعیت اولیه دو عامل مهم در همگرایی الگوریتم ژنتیک هستند. انتخاب جمعیتهای بزرگ موجب کند شدن سرعت همگرایی شده و انتخاب جمعیتهای کوچک باعث می شود که الگوریتم نتواند کل فضای جواب را برای یافتن جواب بهتر جستجو کند. همچنین انتخاب یک جمعیت اولیه مناسب موجب سرعت دادن به همگرایی الگوریتم می شود. ما اندازه جمعیت را $n+p$ در نظر گرفتیم که با بررسیهای انجام شده مناسب به نظر می‌آید. برای انتخاب جمعیت اولیه ما ژنهای ۱، ۲، ...، p را به اولین p تا از اعضای جمعیت، $p+1$، $p+2$، ...، $2p$ را به دومین p تا از اعضای جمعیت اختصاص دادیم و همینطور الی آخر. وقتی اندیس ژنها به n رسید از ۱ شروع کرده و دوتا دوتا به آن اضافه می کنیم، یعنی ۱، ۳، ...، $2p-1$ و همینطور تا ژنهای کل جمعیت تعیین شوند.

۵.۳ انتخاب والدین

برای تولید یک عضو جدید والدین به طور تصادفی از جمعیت انتخاب می شوند. البته روشهای دیگر ازقبیل انتخاب اعضای با مطلوبیت بالاتر به عنوان والدین نیز مورد آزمون قرار گرفت که به نتایج بهتری منجر نشد.

۶.۳ تولید عضو جدید

در الگوریتم ژنتیک کروموزمهای والدین با هم ترکیب می شوند تا یک عضو جدید تولید کنند. معمولا کروموزمهای والدین به دو قسمت شکسته شده و سپس با هم ترکیب می شوند تا دو عضو جدید تولید شود. ما به جای استفاده از این شیوه سنتی از یک روش حریصانه برای تولید عضو جدید استفاده می‌کنیم. به این صورت که ابتدا ژنهایی که در والدین مشترک است را انتخاب می کنیم سپس از بین بقیه ژنهای والدین یکی یکی ژنهایی را انتخاب میکنیم که موجب کمترین افزایش در تابع هدف شوند. این روند را تا رسیدن تعداد ژنها به عدد p ادامه میدهیم. اگر چه این نوع تولید عضو جدید به زمان بیشتری نیاز دارد ولی در عوض موجب بالابردن سطح کیفی جواب می شود.

۷.۳ جهش

انجام عمل جهش در الگوریتم ژنتیک موجب می شود که الگوریتم بتواند از گیر افتادن در نقاط بهینه موضعی فرار کند. برای اجرای عمل جهش، بعد از تولید عضو جدید یکی از ژنهایی که در این عضو جدید قرار ندارد جایگزین یکی از ژنهای درون این عضو جدید می‌شود به گونه‌ای که بیشترین بهبود را برای تابع هدف ایجاد کند.

۸.۳ جایگزینی

بعد از تولید یک عضو و انجام عمل جهش باید این عضو را به جمعیت اضافه کنیم. اگر این عضو همانند هیچکدام از اعضای قبلی جمعیت نباشد و مقدار مطلوبیت آن از بدترین مقدار مطلوبیت جمعیت بهتر باشد آنگاه این عضو وارد جمعیت شده و عضوی از جمعیت که بدترین مقدار مطلوبیت را دارد از جمعیت خارج می شود.

۹.۳ شرط توقف

شرط توقف الگوریتم میتواند رسیدن به حداکثر تعداد تکرار از ابتدای اجرای الگوریتم و یا بعد از آخرین بهبود باشد. همچنین یک حداکثر زمان اجرا را میتوان به عنوان شرط توقف در نظر گرفت. ما رسیدن به $n+p$ تکرار بعد از آخرین بهبود را در نظر گرفتیم.

۱۰.۳ الگوریتم

با توجه به مطالب گفته شده الگوریتم زیر را برای مساله پیشنهاد میکنیم.

Algorithm [GApMP].

Initialization:

Generate an initial population P of size $n+p$ as described in Section 4.3

Find the best member and its fitness value, f_{best} and worst member and its fitness value, f_{worst} in the P.

Let J be the set of all genes.

Iteration counter $r := ۰$, $r_{best} := ۰$.

Iteration step:

While $r - r_{best} \leq n+p$ **do** the following:

1. $r := r + ۱$,
2. Select two members x, y randomly from P. Let J_x and J_y be the genes of x and y, respectively.
3. Generate a new member z:
   (a) Set $J_z = J_x \bigcap J_y$ and compute the fitness value for J_z.
   (b) Find the gene $j \in J_z \setminus (J_x \bigcap J_y)$ so that by adding j to the J_z we have the greatest improvement in the fitness value of J_z.
   (c) If $|J_z| \leq p$ go to step a۳
4. Select two genes one from J_z and one from $J \setminus J_z$, which, when swapped, result in the greatest improvement in the fitness value of J_z.
5. Exchange this two selected genes and compute the fitness value,f_{cur}, of J_z.

6. **If** the generated member is not in the population and $f_{cur} < f_{worst}$ then **do** the following:

   (a) Replace the worst member by the generated member.

   (b) Update the worst member of the population and its fitness value, f_{worst}.

   (c) **If** $f_{cur} < f_{best}$, set $f_{best} = f_{cur}$ and $r_{best} = r$.

endwhile

در بخش بعد ما نتایج الگوریتم GApMP را با روش جستجوی همسایگی متغیر مقایسه می کنیم.

۴ نتایج محاسباتی

برای بررسی عملکرد الگوریتم GApMP از چهل مساله نمونه که در کتابخانه ORLIB [۲] قرار دارد استفاده شد. در این کتابخانه جوابهای تمام چهل مساله برای حالتی که همه رئوس وزن نامنفی دارند وجود دارد. ما برای اینکه از آنها استفاده کنیم ابتدا به پنج راس اول سپس به ده راس اول و در نهایت به رئوس با شماره فرد وزن منفی داده و در هر مورد هر چهل مساله را برای هر دو حالت P_1 و P_2 با استفاده از الگوریتمهای GApMP و جستجوی همسایگی متغیر حل کردیم.

نتایج در جداول پیوست آمده است. در جدول ۲ نتایج حاصل از الگوریتم GApMP برای حالتی که همه رئوس دارای وزن مثبت دارند آمده است که می توان نتایج را با جوابهای واقعی مقایسه کرد. در این جدول ستون با عنوان Last improv. iteration نشان دهنده شماره آخرین تکراری است که بعد از آن دیگر تابع هدف بهبود نیافته است. ستون با عنوان $\%Error$ بیان کننده درصد خطای نسبی یعنی

$$(\frac{f - f_{opt}}{|f_{opt}|}) \times ۱۰۰$$

برای بهترین و بدترین جواب در جمعیت است.

جدل ۳ میانگین زمان اجرای الگوریتمهای GApMP و جستجوی همسایگی متغیر را برای مدلهای P_1 و P_2 نشان میدهد. جداول ۴ تا ۹ نتایج بدست آمده از هر الگوریتم، برای حالاتی که رئوس دارای وزن منفی هستند را در بر دارند. ستون با عنوان Best در این

جداول بیان کننده بهترین جواب بدست آمده برای مساله از بین دو روش است و ستون با عنوان $\%Error$ بیان کننده درصد خطای نسبی هر روش نسبت به بهترین جواب می باشد.

مشاهده می شود که از بین ۲۴۰ مساله حل شده، روش GApMP در ۲۲۷ مورد به بهترین جواب رسیده است و روش جستجوی همسایگی متغیر در ۱۱۱ مورد. همچنین مجموع درصد خطاها برای GApMP، ۱۲/۵ و برای جستجوی همسایگی متغیر ۹۷/۰۶ است که دلالت بر بهتر بودن روش GApMP دارد.

۵ سایر الگوریتمهای ژنتیک برای مساله p—میانه

اولین الگوریتم ژنتیک برای مساله p—میانه توسط هاسج و گودچایلد[۲۶] [۱۹] ارائه شد. در الگوریتم آنها هر جواب با یک رشته دودویی شامل n عضو (ژن) نمایش داده می شود که هر عضو با اندیس سرویس دهنده متناظر است. هر ژن نشان میدهد که آیا سرویس دهنده متناظر با آن به عنوان میانه انتخاب شده است یا نه (۰ یا ۱). کدگذاری آنها به گونه‌ای است که تضمینی برای اینکه دقیقاً p میانه انتخاب شود وجود ندارد بنابراین آنها از یک تابع جریمه برای اعمال این محدودیت استفاده می کنند. با توجه به این نوع کدگذاری نتایج بدست آمده از الگوریتم آنها نتایج زیاد خوبی نمی‌باشد.

دیبل و دشام [۲۷] [۱۰] الگوریتمی با کدگذاری مناسبتری ارائه داده‌اند. در الگوریتم آنها هر عضو از جمعیت دقیقاً p ژن دارد که هر ژن نشان دهنده اندیس یک سرویس دهنده است. آنها الگوریتمشان را برای حل یک مساله با $n = ۱۵۰$ و $p = ۹$ به کار بردند و نتایج را با نتایج بدست آمده از روش جایگزینی ابتکاری که مربوط به تیتز و بارتز[۲۸] [۳۰] است مقایسه کردند. با وجود اینکه روش ژنتیک آنها زمان بیشتری را می گیرد ولی نتایج بهتری ارائه نمی دهد.

مورنو پرز[۲۹] و همکاران [۲۲] یک الگوریتم ژنتیک موازی ارائه کرده‌اند که در آن گروههای جمعیتی چندگانه وجود دارند و اعضای این گروهها با یکدیگر رد و بدل می شوند. آنها نتایج محاسباتی روش خود را ارائه نکرده اند.

بزکایا[۳۰] و همکاران [۴] نیز یک اللگوریتم ژنتیک همراه با پارامترهای مختلف ارائه کرده‌اند. نتایج الگوریتم آنها از نتایج روش جایگزینی بهتر می باشد.

---

[۲۶] Hosage and Goodchild

[۲۷] Dibble and Densham

[۲۸] Teitz and Bart

[۲۹] Moreno-Perez

[۳۰] Bozkaya

یکی از الگوریتمهایی که اخیراً برای مساله p–میانه ارائه شده است الگوریتم آلپ[۳۱] و همکاران [۱] است. آنها از یک الگوریتم حریصانه کاهشی برای تولید عضو جدید استفاده می کنند. برای انجام این کار ابتدا کرموزمهای والدین ادغام شده سپس یکی یکی ژنهایی که با خروج آنها بهترین تابع مطلوبیت بدست می آید از این مجموعه خارج می شوند تا تعداد کرموزمها به p برسد. در مقایسه با روش آنها ما روش حریصانه افزایشی را برای تولید عضو جدید به کار برده‌ایم. همچنین ما یک گام از روش جایگزینی را به عنوان عمل جهش استفاده کرده ایم ولی آلپ و همکاران از جهش استفاده نکرده‌اند.

٦ خلاصه و نتیجه گیری

در این طرح ما یک الگوریتم ژنتیک برای مساله p–میانه روی شبکه های با وزن مثبت و منفی ارائه کرده و نتایج حاصل از آن را با نتایج به دست آمده از روش جستجوی همسایگی متغیر مقایسه کردیم. با توجه به نتایج به دست آمده مشاهده می شود روش ژنتیکی که ما ارائه کرده‌ایم در اکثر موارد جواب بهتری از روش جستجوی همسایگی متغیر بدست می دهد.

مراجع

[1] Alp, O., Erkut E. and Drezner Z. (2003), "An efficient genetic algorithm for the p-Median Problem." *Annals of Operations Research*, 122: 21-42.

[2] Beasley J.E. (1990), "OR-Library: distributing test problems by electronic mail." *Journal of the Operational Research Society*, 41(11): 1069-1072.

[3] Beasley J.E. (1993), "Lagrangean heuristics for location problems." *European Journal of Operational Research*, 65: 383-399.

[۳۱] Alp

[4] Bozkaya, Zhang J. and Erkut E. (2002), "A genetic algorithm for the p-Median Problem." In Z. Drezner and H. Hamacher (eds.), *Facility Location: Applications and Theory.* Berlin: Springer.

[5] Burkard R.E. and Krarup J. (1998), "A linear algorithm for the pos/neg-weighted 1-median problem on a cactus." *Computing*, 60: 193-215.

[6] Burkard R.E., Çela E. and Dollani H. (2000), "2-median in trees with pos/neg weights." *Discrete Appl. Math.*, 105: 51-71.

[7] Burkard R.E., Fathali J. (2004), "A polynomial time method for the pos/neg weighted 3-median problem on a tree." *SFB Report* 333, Institute of Mathematics B, Graz University of Technology.

[8] Cappanera P. (1999), "A survey on obnoxious facility location problems." *Technical Report:* TR-99-11, Dipartimento di Informatica, Universita di Pisa.

[9] Carrizosa E. and Plastria F. (1999), "Location of semi-obnoxious facilities." *Stud. Locational Anal.*, 12: 1-27.

[10] Dibble C. and Densham P.J. (1993), "Generating interesting alternatives in GIS and SDSS using genetic algorithms." *GIS/LIS Symposium*, University of Nebraska, Lincoln.

[11] Captivo E.M. (1991), "Fast primal and dual heuristics for the p-median location problem." *European Journal of Operational Research*, 52: 65-74.

[12] Eiselt H.A. and Laporte G. (1995), "Objectives in location problems." Chapter 8 in: *Facility Location. A Survey of Applications and Methods.* Z.Drezner (eds.), Springer-Verlag, New York, Inc.

[13] Erlenkotter D. (1978), "A dual based procedure for uncapacitated facility location." *Operations Research*, 26: 992-1009.

[14] Fathali J. and Kakhki H.T. (2006), "Solving the p-median problem with pos/neg weights by variable neighborhood search and some

results for special cases." *European Journal of Operational Research,* 170: 440-462.

[15] , Fathali J., Kakhki H.T. and Burkard R.E., "An ant colony algorithm for the pos/neg weighted p-median problem." To apear in *Central European Journal of Operations Research.*

[16] Galvao R.D. (1980), "A dual bounded algorithm for the p-median problem." *Operations Research,* 28: 190-206.

[17] Goldberg D.E. (1989), *Genetic Algorithms In Search, Optimization And Machine Learning.* Menlo Park, CA: Addison-Weseley.

[18] Hansen P. and Mladenovic N. (1997), "Variable neighborhood search for the p-median." *Location Sci.*, 5: 207-226.

[19] Hosage M. and Goodchild M.F. (1986), "Discrete space location-allocation solutions from genetic algorithms." *Annals of Operational Research*, 6: 35-46.

[20] Kariv O, and Hakimi S. L. (1979), "An algorithmic approach to network location problems. Part II: p-medians." *SIAM J. Appl. Math.*, 37: 539-560.

[21] Kuehn A. A. and Hamburger M.J. (1963), "A heuristic program for locating warehouses." *Management Sci.*, 9: 643-666.

[22] Moreno-Perez J.A., Moreno-Vega J.M. and Mladenovic N. (1994), "Tabu search and simulated annealing in p-median problems." *Talk at the Canadian Operational Research Society Conference, Montral.*

[23] Plastria F. (1996), "Optimal location of undesirable facilities: an overview." *Belg. J. Oper. Res. Statist. Comput. Sci.* 36: 109-127.

[24] Resende M.G.C. and Werneck R. (2004), "A Hybrid heuristic for the p-median problem." *Journal of Heuristics*, 10: 59-88.

[25] Reeves C.R. (1993), "Genetic Algorithms." In C.R Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems.* Chapter 4, pp. 151-196.

[26] ReVelle C.S. and Swain R. (1970), "Centeral facilities location." *Geographical Analysis,* 2: 30-42.

[27] Rolland E., Schilling D.A. and Current J. R. (1996), "An efficient tabu search heuristic for the p-median problem." *European Journal of Operational Research,* 96: 329-342.

[28] Senne E.L. and Lorena L.A.N. (2000), "Lagrangean/surrogate heuristics for p-median problems." Chapter 3 in: *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research,* M.Laguna and J.L.Gonzalez-velarde (eds.), Kluwer Academic Publishers.

[29] Tamir A. (1996), "An $o(pn^2)$ algorithm for the p-median and related problems on tree graphs." *Operations Research Letters,* 19: 59-64.

[30] Teitz M.B. and Bart P. (1968), "Heuristic methods for estimating generalized vertex median of a weighted graph." *Operations Research,* 16: 955-961.

[31] Whitaker R. (1983), "A fast algorithm for the greedy interchange for large-scale clustering and median location problems." *INFOR,* 21: 95-108.

پیوست ۱

جداول

Test #	n	p	Optimal	GA	Last improv. iteration	%Error Best	%Error Worst	CPU time (sec)
1	100	5	5819	5819	61	0.00	46.64	0.35
2	100	10	4093	4093	148	0.00	17.43	0.65
3	100	10	4250	4250	170	0.00	13.49	0.70
4	100	20	3034	3034	214	0.00	9.53	0.85
5	100	33	1355	1355	204	0.00	8.63	0.95
6	200	5	7824	7824	92	0.00	15.74	0.80
7	200	10	5631	5631	156	0.00	7.55	1.50
8	200	20	4445	4445	295	0.00	2.50	3.33
9	200	40	2734	2739	404	0.18	0.80	5.00
10	200	67	1255	1260	311	0.40	2.34	6.33
11	300	5	7696	7696	160	0.00	13.11	6.33
12	300	10	6634	6634	312	0.00	6.61	9.63
13	300	30	4374	4374	552	0.00	1.47	17.33
14	300	60	2968	2969	722	0.03	0.54	30.33
15	300	100	1729	1736	618	0.40	0.64	36.00
16	400	5	8162	8162	175	0.00	16.59	19.66
17	400	10	6999	6999	402	0.00	6.60	32.33
18	400	40	4809	4811	722	0.04	0.76	58.33
19	400	80	2845	2849	1365	0.14	0.37	104.00
20	400	133	1789	1789	1700	0.00	0.45	148.33
21	500	5	9138	9138	150	0.00	25.68	41.33
22	500	10	8579	8579	633	0.00	4.20	70.33
23	500	50	4619	4619	1047	0.00	0.87	141.00
24	500	100	2961	2961	1627	0.00	0.36	253.33
25	500	167	1828	1831	1531	0.16	0.47	391.66
26	600	5	9917	9917	260	0.00	16.26	89.33
27	600	10	8307	8307	746	0.00	3.76	138.66
28	600	60	4498	4498	1758	0.00	0.31	311.66
29	600	120	3033	3034	1720	0.03	0.38	568.00
30	600	200	1989	1995	1932	0.30	0.65	957.66
31	700	5	10086	10086	265	0.00	21.13	152.00
32	700	10	9297	9297	760	0.00	7.01	188.00
33	700	70	4700	4703	2113	0.06	0.40	543.66
34	700	140	3013	3013	2081	0.00	0.33	1120.00
35	800	5	10400	10400	349	0.00	17.46	216.66
36	800	10	9934	9934	811	0.00	4.45	302.33
37	800	80	5057	5058	2586	0.02	0.31	1111.33
38	900	5	11060	11060	436	0.00	14.41	316.33
39	900	10	9423	9423	756	0.00	6.96	487.66
40	900	90	5128	5129	3793	0.02	0.26	2120.00

جدول ۲: نتایج حاصل از الگوریتم GApMP برای مسائل با وزن مثبت.

			P_1		P_2	
Test#	n	p	GA	MVNS	GA	MVNS
1	100	5	1.000	1.000	1.000	0.400
2	100	10	1.550	1.400	1.500	1.200
3	100	10	1.333	1.400	1.500	1.200
4	100	20	1.777	1.800	1.777	1.800
5	100	33	1.777	2.200	1.555	2.600
6	200	5	2.666	4.000	2.444	3.000
7	200	10	3.000	4.600	2.888	4.600
8	200	20	4.777	10.400	4.111	8.400
9	200	40	7.222	18.000	5.777	17.600
10	200	67	8.222	30.200	6.777	26.000
11	300	5	9.777	12.400	7.777	9.000
12	300	10	11.888	20.600	11.666	18.400
13	300	30	21.777	69.400	18.444	55.800
14	300	60	38.625	118.400	36.125	93.800
15	300	100	35.777	169.400	32.444	138.800
16	400	5	25.333	29.600	24.777	25.000
17	400	10	41.500	47.800	37.000	40.200
18	400	40	63.888	200.200	58.111	174.600
19	400	80	105.777	367.800	86.333	302.600
20	400	133	151.555	545.000	117.888	444.800
21	500	5	49.777	37.200	43.375	29.200
22	500	10	80.777	82.400	75.666	70.000
23	500	50	143.333	432.000	127.111	365.000
24	500	100	242.000	764.800	218.222	621.200
25	500	167	363.222	1189.600	279.444	978.600
26	600	5	102.000	68.200	102.222	60.8
27	600	10	165.666	161.200	141.666	133.400
28	600	60	336.777	825.600	284.888	707.200
29	600	120	540.444	1451.200	531.888	1229.000
30	600	200	817.111	2142.200	734.000	1812.800
31	700	5	162.777	74.000	141.429	67.000
32	700	10	246.333	183.200	201.111	166.200
33	700	70	607.222	1354.800	558.555	1114.600
34	700	140	1083.555	2411.800	1038.444	1948.800
35	800	5	241.888	119.600	217.222	98.200
36	800	10	395.444	237.400	379.777	219.800
37	800	80	1125.666	1944.600	985.111	1624.200
38	900	5	354.222	215.200	309.333	173.000
39	900	10	474.777	299.000	428.333	261.800
40	900	90	2051.555	2873.800	1612.888	2301.600

جدول ۳: میانگین زمان اجرا برای مدلهای P_1 و P_2.

			Objective Function Value			%Error	
Test #	n	p	Best	GA	MVNS	GA	MVNS
1	100	5	4681	4730	4681	1.05	0.00
2	100	10	2915	2915	2915	0.00	0.00
3	100	10	2529	2529	2529	0.00	0.00
4	100	20	1432	1432	1450	0.00	1.26
5	100	33	61	61	63	0.00	3.17
6	200	5	7061	7061	7061	0.00	0.00
7	200	10	4812	4846	4812	0.71	0.00
8	200	20	3399	3399	3419	0.00	0.59
9	200	40	1919	1919	1926	0.00	0.36
10	200	67	514	514	522	0.00	1.56
11	300	5	7290	7290	7290	0.00	0.00
12	300	10	6201	6201	6201	0.00	0.00
13	300	30	3798	3798	3798	0.00	0.00
14	300	60	2269	2269	2283	0.00	0.62
15	300	100	1153	1153	1157	0.00	0.35
16	400	5	7787	7787	7801	0.00	0.18
17	400	10	6723	6735	6723	0.00	0.18
18	400	40	4219	4219	4219	0.00	0.00
19	400	80	2420	2420	2420	0.00	0.00
20	400	133	1346	1346	1355	0.00	0.67
21	500	5	8888	8888	8888	0.00	0.00
22	500	10	8230	8230	8304	0.00	0.90
23	500	50	4256	4256	4256	0.00	0.00
24	500	100	2538	2538	2538	0.00	0.00
25	500	167	1394	1394	1405	0.00	0.79
26	600	5	9655	9655	9655	0.00	0.00
27	600	10	8038	8038	8040	0.00	0.02
28	600	60	4043	4043	4043	0.00	0.00
29	600	120	2698	2698	2703	0.00	0.19
30	600	200	1604	1604	1614	0.00	0.62
31	700	5	9909	9909	9910	0.00	0.04
32	700	10	8935	8955	8935	0.22	0.00
33	700	70	4411	4411	4417	0.00	0.14
34	700	140	2605	2605	2620	0.00	0.57
35	800	5	10230	10230	10230	0.00	0.00
36	800	10	9735	9735	9767	0.00	0.33
37	800	80	4723	4723	4728	0.00	0.11
38	900	5	10860	10860	10861	0.00	0.01
39	900	10	9070	9070	9070	0.00	0.00
40	900	90	4862	4862	4863	0.00	0.02
Total						1.98	12.68

جدول ۴: نتایج حاصل از الگوریتم GApMP برای مدل P_1 وقتی تنها ۵ راس وزن منفی دارند.

			Objective Function Value			%Error	
Test #	n	p	Best	GA	MVNS	GA	MVNS
1	100	5	3611	3611	3611	0.00	0.00
2	100	10	1247	1247	1247	0.00	0.00
3	100	10	1029	1029	1029	0.00	0.00
4	100	20	-52	-52	-49	0.00	5.77
5	100	33	-1143	-1143	-1143	0.00	0.00
6	200	5	6374	6374	6374	0.00	0.00
7	200	10	4095	4095	4095	0.00	0.00
8	200	20	2575	2575	2575	0.00	0.00
9	200	40	1088	1088	1091	0.00	0.28
10	200	67	-204	-204	-196	0.00	3.92
11	300	5	6756	6756	6756	0.00	0.00
12	300	10	5610	5610	5610	0.00	0.00
13	300	30	3193	3193	3193	0.00	0.00
14	300	60	1480	1480	1489	0.00	0.61
15	300	100	635	635	638	0.00	0.47
16	400	5	7426	7426	7426	0.00	0.00
17	400	10	6292	6292	6301	0.00	0.14
18	400	40	3693	3693	3694	0.00	0.03
19	400	80	2013	2013	2016	0.00	0.15
20	400	133	910	910	919	0.00	0.98
21	500	5	8582	8630	8582	0.56	0.00
22	500	10	7765	7765	7775	0.00	0.13
23	500	50	3795	3795	3795	0.00	0.00
24	500	100	2151	2151	2161	0.00	0.46
25	500	167	990	990	1002	0.00	1.21
26	600	5	9400	9400	9400	0.00	0.00
27	600	10	7651	7651	7651	0.00	0.00
28	600	60	3576	3576	3582	0.00	0.17
29	600	120	2359	2359	2363	0.00	0.17
30	600	200	1199	1199	1210	0.00	0.92
31	700	5	9519	9688	9519	1.78	0.00
32	700	10	8362	8418	8362	0.67	0.00
33	700	70	4144	4144	4153	0.00	0.17
34	700	140	2219	2219	2233	0.00	0.63
35	800	5	10039	10039	10039	0.00	0.00
36	800	10	9415	9415	9442	0.00	0.27
37	800	80	4384	4384	4395	0.00	0.25
38	900	5	10696	10696	10754	0.00	0.54
39	900	10	8535	8535	8535	0.00	0.00
40	900	90	4695	4595	4606	0.00	0.24
Total						3.01	17.51

جدول ۵: نتایج حاصل از الگوریتم GApMP برای مدل P_1 وقتی تنها ۱۰ راس وزن منفی دارند.

			Objective Function Value			%Error	
Test #	n	p	Best	GA	MVNS	GA	MVNS
1	100	5	-7651	-7651	-7651	0.00	0.00
2	100	10	-9445	-9445	-9445	0.00	0.00
3	100	10	-12398	-12398	-12398	0.00	0.00
4	100	20	-11507	-11507	-11507	0.00	0.00
5	100	33	-10811	-10811	-10811	0.00	0.00
6	200	5	-9971	- 9971	-9971	0.00	0.00
7	200	10	-10403	-10403	-10402	0.00	0.01
8	200	20	-13912	-13912	-13901	0.00	0.08
9	200	40	-13997	-13997	-13988	0.00	0.06
10	200	67	-12437	-12437	-12420	0.00	0.14
11	300	5	-10271	-10271	-10271	0.00	0.00
12	300	10	-14850	-14850	-14846	0.00	0.03
13	300	30	-13557	-13557	-13553	0.00	0.03
14	300	60	-17676	-17676	-17674	0.00	0.01
15	300	100	-14437	-14437	-14433	0.00	0.03
16	400	5	-10792	-10792	-10792	0.00	0.00
17	400	10	-11583	-11583	-11583	0.00	0.00
18	400	40	-16286	-16286	-16282	0.00	0.02
19	400	80	-14200	-14200	-14195	0.00	0.04
20	400	133	-16362	-16362	-16355	0.00	0.04
21	500	5	-11296	-11296	-11296	0.00	0.00
22	500	10	-16588	-16588	-16588	0.00	0.00
23	500	50	-15272	-15272	-15257	0.00	0.10
24	500	100	-17221	-17221	-17207	0.00	0.08
25	500	167	-17924	-17924	-17919	0.00	0.03
26	600	5	-13060	-13060	-13060	0.00	0.00
27	600	10	-16204	-16204	-16179	0.00	0.15
28	600	60	-22970	-22970	-22969	0.00	0.01
29	600	120	-17796	-17796	-17793	0.00	0.02
30	600	200	-21333	-21333	-21324	0.00	0.04
31	700	5	-11466	-11396	-11466	0.61	0.00
32	700	10	-30465	-30465	-30454	0.00	0.04
33	700	70	-16914	-16914	-16908	0.00	0.04
34	700	140	-23803	-23803	-23795	0.00	0.03
35	800	5	-14709	-14709	-14709	0.00	0.00
36	800	10	-21934	-21934	-21934	0.00	0.00
37	800	80	-21038	-21038	-21034	0.00	0.02
38	900	5	-21059	-21059	-21059	0.00	0.00
39	900	10	-38980	-38980	-38980	0.00	0.00
40	900	90	-19350	-19350	-19345	0.00	0.03
Total						0.61	1.08

جدول ٦: نتایج حاصل از الگوریتم GApMP برای مدل P_1 وقتی نیمی از رئوس وزن منفی دارند.

			Objective Function Value			%Error	
Test #	n	p	Best	GA	MVNS	GA	MVNS
1	100	5	5324	5324	5324	0.00	0.00
2	100	10	3696	3696	3709	0.00	0.35
3	100	10	3592	3592	3592	0.00	0.00
4	100	20	2460	2460	2465	0.00	0.20
5	100	33	994	994	1010	0.00	1.61
6	200	5	7266	7266	7266	0.00	0.00
7	200	10	5224	5224	5238	0.00	0.27
8	200	20	4034	4034	4034	0.00	0.00
9	200	40	2540	2540	2547	0.00	0.28
10	200	67	1077	1077	1081	0.00	0.37
11	300	5	7440	7440	7440	0.00	0.00
12	300	10	6511	6511	6511	0.00	0.00
13	300	30	4187	4187	4190	0.00	0.07
14	300	60	2781	2785	2781	0.14	0.00
15	300	100	1581	1581	1590	0.00	0.57
16	400	5	7870	7870	7870	0.00	0.00
17	400	10	6849	6849	6853	0.06	0.00
18	400	40	4589	4589	4589	0.00	0.00
19	400	80	2741	2741	2749	0.00	0.29
20	400	133	1703	1703	1709	0.00	0.35
21	500	5	8980	8980	8980	0.00	0.00
22	500	10	8403	8403	8471	0.00	0.81
23	500	50	4520	4520	4520	0.00	0.00
24	500	100	2853	2853	2857	0.00	0.14
25	500	167	1735	1735	1748	0.00	0.75
26	600	5	9719	9719	9719	0.00	0.00
27	600	10	8137	8137	8137	0.00	0.00
28	600	60	4385	4385	4385	0.00	0.00
29	600	120	2965	2965	2977	0.00	0.40
30	600	200	1939	1939	1947	0.00	0.62
31	700	5	9915	9968	9915	0.53	0.00
32	700	10	9179	9179	9179	0.00	0.00
33	700	70	4623	4624	4623	0.02	0.00
34	700	140	2913	2913	2922	0.00	0.31
35	800	5	10286	10286	10286	0.00	0.00
36	800	10	9803	9803	9815	0.00	0.12
37	800	80	4967	4967	4972	0.00	0.10
38	900	5	10914	10914	10914	0.00	0.00
39	900	10	9305	9305	9305	0.00	0.00
40	900	90	5075	5075	5102	0.00	0.53
Total						0.75	8.14

جدول ۷: نتایج حاصل از الگوریتم GApMP برای مدل P_2 وقتی تنها ۵ راس وزن منفی دارند.

			Objective Function Value			%Error	
Test #	n	p	Best	GA	MVNS	GA	MVNS
1	100	5	4826	4826	4826	0.00	0.00
2	100	10	2976	2976	2976	0.00	0.00
3	100	10	3341	3341	3341	0.00	0.00
4	100	20	1854	1854	1854	0.00	0.00
5	100	33	533	533	567	0.00	6.75
6	200	5	6787	6787	6787	0.00	0.00
7	200	10	4949	4949	4949	0.00	0.00
8	200	20	3812	3812	3820	0.00	0.21
9	200	40	2391	2391	2393	0.00	0.08
10	200	67	904	904	909	0.00	0.55
11	300	5	7136	7136	7136	0.00	0.00
12	300	10	6395	6395	6395	0.00	0.00
13	300	30	4011	4011	4011	0.00	0.00
14	300	60	2564	2564	2564	0.00	0.00
15	300	100	1462	1462	1471	0.00	0.62
16	400	5	7624	7624	7624	0.00	0.00
17	400	10	6668	6668	6669	0.00	0.01
18	400	40	4437	4437	4437	0.00	0.00
19	400	80	2633	2633	2634	0.00	0.04
20	400	133	1623	1623	1633	0.00	0.62
21	500	5	8800	8800	8800	0.00	0.00
22	500	10	8291	8291	8360	0.00	0.83
23	500	50	4337	4337	4339	0.00	0.05
24	500	100	2748	2779	2748	0.84	0.00
25	500	167	1636	1636	1644	0.00	0.49
26	600	5	9528	9528	9572	0.00	0.46
27	600	10	8004	8004	8004	0.00	0.00
28	600	60	4296	4296	4296	0.00	0.00
29	600	120	2897	2897	2902	0.00	0.17
30	600	200	1850	1850	1858	0.00	0.43
31	700	5	9820	9820	9820	0.00	0.00
32	700	10	9053	9053	9053	0.00	0.00
33	700	70	4572	4572	4583	0.00	0.24
34	700	140	2835	2835	2852	0.00	0.60
35	800	5	10142	10142	10142	0.00	0.00
36	800	10	9637	9637	9637	0.00	0.00
37	800	80	4875	4875	4884	0.00	0.18
38	900	5	10800	10800	10839	0.00	0.36
39	900	10	9201	9201	9201	0.00	0.00
40	900	90	5026	5026	5044	0.00	0.36
Total						0.84	13.05

جدول ۸: نتایج حاصل از الگوریتم GApMP برای مدل P_2 وقتی تنها ۱۰ راس وزن منفی دارند.

			Objective Function Value			%Error	
Test #	n	p	Best	GA	MVNS	GA	MVNS
1	100	5	-635	-635	-635	0.00	0.00
2	100	10	-1245	-1245	-1245	0.00	0.00
3	100	10	-1131	-1131	-1131	0.00	0.00
4	100	20	-1477	-1477	-1477	0.00	0.00
5	100	33	-1687	-1687	-1687	0.00	0.00
6	200	5	-1163	-1163	-1135	0.00	2.41
7	200	10	-1360	-1360	-1360	0.00	0.00
8	200	20	-1765	-1765	-1764	0.00	0.06
9	200	40	-2212	-2212	-2212	0.00	0.00
10	200	67	-1815	-1815	-1788	0.00	1.49
11	300	5	-797	-797	-797	0.00	0.00
12	300	10	-1290	-1290	-1285	0.00	0.39
13	300	30	-1709	-1709	-1674	0.00	2.05
14	300	60	-2224	-2224	-2190	0.00	1.53
15	300	100	-2152	-2152	-2151	0.00	0.05
16	400	5	-932	-932	-840	0.00	9.87
17	400	10	-1318	-1254	-1318	4.86	0.00
18	400	40	-2096	-2096	-2096	0.00	0.00
19	400	80	-2119	-2119	-2116	0.00	0.14
20	400	133	-2291	-2291	-2277	0.00	0.61
21	500	5	-687	-687	-687	0.00	0.00
22	500	10	-1111	-1111	-1098	0.00	1.17
23	500	50	-1933	-1933	-1915	0.00	0.93
24	500	100	-2216	-2216	-2186	0.00	1.35
25	500	167	-2376	-2376	-2357	0.00	0.80
26	600	5	-820	-820	-808	0.00	1.46
27	600	10	-1053	-1053	-1053	0.00	0.00
28	600	60	-2119	-2119	-2107	0.00	0.57
29	600	120	-2198	-2198	-2192	0.00	0.27
30	600	200	-2320	-2320	-2279	0.00	1.77
31	700	5	-748	-748	-741	0.00	0.94
32	700	10	-1030	-1030	-934	0.00	9.32
33	700	70	-2009	-2009	-1976	0.00	1.65
34	700	140	-2436	-2436	-2423	0.00	0.54
35	800	5	-855	-855	-855	0.00	0.00
36	800	10	-985	-985	-985	0.00	0.00
37	800	80	-2278	-2278	-2262	0.00	0.70
38	900	5	-607	-607	-607	0.00	0.00
39	900	10	-901	-901	-860	0.00	4.55
40	900	90	-2347	-2347	-2346	0.00	0.04
Total						4.86	44.60

جدول ۹: نتایج حاصل از الگوریتم GApMP برای مدل P_2 وقتی نیمی از رئوس وزن منفی دارند.

پیوست ۲

مقاله مستخرج از طرح پژوهشی

Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 183 (2006) 1071–1083
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

A genetic algorithm for the p-median problem with pos/neg weights

Jafar Fathali

Department of Mathematics, Shahrood University of Technology, University Blvd., Shahrood, Iran

Abstract

Let a connected undirected graph $G = (V, E)$ be given. In the classical p-median problem we want to find a set X containing p points in G such that the sum of weighted distances from X to all vertices in V is minimized. We consider the semi-obnoxious case where every vertex has either a positive or negative weight. In this case we have two different objective functions: the sum of the minimum weighted distances from X to all vertices and the sum of the weighted minimum distances. In this paper we propose a genetic algorithm for both problems. Computational results are compared with a previously investigated variable neighborhood search algorithm.

Keywords: Genetic algorithm; Location theory; p-Median problem; Obnoxious facilities

1. Introduction

In discrete location theory, p-median problems play a central role. In the classic p-median problem the goal is to select the locations of p facilities to serve n demand points so that the sum of the distances from facilities to demand points is minimized. Usually to every demand point v_i is also assigned a weight w_i, so the problem actually asks for the minimum weighted sum, i.e.:

$$\min f(X) = \sum_{i=1}^{n} w_i d(X, v_i), \tag{1}$$

where X is the set of the facilities and $d(X, v)$ is defined as the shortest distance from v to the facility $x \in X$ closest to v.

Kariv and Hakimi [16] showed that the p-median problem is $\mathcal{NP}$-hard; hence, there is a special need for good heuristics. Among the first heuristics proposed for the classic p-median problem are the greedy method of Kuehn and Hamburger [17] as well as the interchange method of Teitz and Bart [23]. Other heuristics proposed for the p-median problem are the fast interchange heuristic [24], tabu search (e.g. [22]), variable

E-mail address: fathali@shahrood.ac.ir

0096-3003/$ - see front matter
doi:10.1016/j.amc.2006.05.143

neighborhood search [14], genetic algorithms (we discuss genetic algorithms for the p-median problem in Section 4) and hybrid heuristic methods [20].

If the facilities are obnoxious for a part of the clients, then a location problem with positive and negative weights can be considered. Cappanera [8] discussed and classified many obnoxious location problems. For further surveys on (semi-)obnoxious location problems, see Carrizosa and Plastria [9], Eiselt and Laporte [11] and Plastria [19].

In networks, the first location model with positive and negative weights is due to Burkard and Krarup [4], who showed that the 1-median problem in a cactus with positive and negative vertex weights can be solved in linear time. Burkard et al. [5] considered p-median problems with pos/neg weights in graphs and noted that there are two different models:

In the first model (P1) the sum of the *minimum weighted distances* is minimized:

$$f_1(X) = \sum_{i=1}^{n} \min_{1 \leqslant j \leqslant p} (w_i d(v_i, x_j)). \tag{P1}$$

In the second model (P2) the sum of the *weighted minimum distances* is minimized:

$$f_2(X) = \sum_{i=1}^{n} w_i \min_{1 \leqslant j \leqslant p} d(v_i, x_j). \tag{P2}$$

For problem (P1), Burkard et al. [5] exploit some structural properties and develop an $O(n^2)$ algorithm for finding the 2-median in a tree with n vertices, an $O(n \log n)$ algorithm for stars, and a linear algorithm for paths. For problem (P2) they present an $O(n^3)$ algorithm for finding the 2-median in a tree. If the medians are restricted to vertices, then the complexity is reduced to $O(n^2)$. Burkard and Fathali [7] extend the algorithm for 3-median in a tree for model (P2).

Some heurisitc methods are proposed for the positive and negative weighted p-median problems. Fathali and Kakhki [12] applied a modified version of the variable neighborhood search and Burkard et al. [6] presented an ant colony algorithm to this problem. In this paper we develop a genetic algorithm for the pos/neg weighted p-median problem.

In what follows we describe our genetic algorithm for the p-median problem with positive and negative weights in Section 2. Computational results are presented in Section 3. In Section 4 related works are discussed. Section 5 contains a summary and conclusions.

2. The proposed genetic algorithm

In this section we present a genetic algorithm for the positive and negative weighted p-median problem. This algorithm can be applied to both models, (P1) and (P2).

2.1. Genetic algorithms

Genetic algorithms (GAs) are search heuristic methods for solving combinatorial optimization problems. They begin with a feasible solution and seek to improve upon it. GAs invented by John Holland in the 196Os, but they have become popular in the operations research literature more recently.

In the GAs each chromosome corresponds to a solution for the problem. The measure of quality of a solution is called fitness. A genetic operator called crossover is used to produce new chromosomes from a pair of selected chromosomes. Mutations are used to promote genetic diversity. For more details about genetic algorithms the reader is referred to the books by Goldberg [13] and Reeves [21].

2.2. Encoding

In our encoding each chromosome has exactly p genes where each genes corresponds to the index of a selected facility. For example, $\{7, 5, 2, 10, 14\}$ is a chromosome corresponds to a feasible solution for a problem with 5 median where demand points $2, 5, 7, 10$ and 14 are selected as location of facilities.

2.3. Fitness evaluation

The fitness of a chromosome is given by the objective function value of the corresponding solution for the considered p-median problems. To calculate the objective function value of models (P1) and (P2) one should note that when some points have negative weights the manner of assignment demand points to the facilities differ. In model (P1) the vertices with nonnegative weights are assigned to the closest facility and the vertices with negative weights are assigned to the farthest facility. But in model (P2) both positively and negatively weighted vertices are assigned to the closest facility.

2.4. Population

The population size and the initial population are two effective factors on the convergence of the algorithm. Large populations slow down the GA while small population may not have sufficient genetic diversity to search over the feasible region. In our algorithm we set $n + p$ as the population size. Selection a suitable initial population results fast convergence for the algorithm. To achieve a near optimal solution the initial population should be distributed on the whole feasible region and every gene must be presented in the initial population. To select members of initial population, we assign the genes $1, 2, \ldots, p$ to the first member, the genes $p + 1, p + 2, \ldots, 2p$ to the second member and so on. When the gene index exceeds n we start from 1 with increment of two in the sequence, i.e. the first member of this group is $1, 3, 5, \ldots, 2p - 1$. Similarly, in the next groups we use increments of three, four, five and so on.

2.5. Selection

To generate new members, we select the parents randomly from the population. Other methods such as selecting members with better fitness as the parents can be used. However, our computational experiments showed that this method did not yield better results.

2.6. Generating new members

In a GA the chromosomes of two parents are merged to generate new members (children). Usually by using a crossover the chromosomes of the parents are split into two part and then combined to generate two new members. Instead of this traditional crossover operation we use a greedy-add heuristic method to generate a new member. We select the genes that are present in both parents, as a part of genes of the new member. Then among remained genes in the chromosomes of parents, one at a time is added to the genes of the new member until the number of genes reaches p. To increase the number of genes by one we add the gene whose yields the smallest increase in the objective function value. This children generation method increases time complexity however it improves the quality of the generated solutions.

2.7. Mutation

The mutation operator helps the GA to escape from local optimum. To perform mutation, after generating a new member, we replace one gene in this new member by a gene which is not in the new member so that the fitness has best improvement. This is equivalent to one step of interchange heuristic method. In the interchange heuristic method this procedure is repeated until no better solution can be found. The most efficient implementation of this method was presented by Whitaker [24]. To perform this operation for models (P1) and (P2) we use one step of adjusted fast interchange heuristic method which is adjusted Whitaker's algorithm for pos/neg weighted p-median problem and can be found in [12]. Note that this is a very computationally expensive operator, since each time it is applied, a large number of fitness function needs to be performed. However, the quality of the new solutions will be improved.

2.8. Replacement

After generate a member and implement the mutation operator, we should admit this member to the population. If the new member is not identical to an existing member and its fitness value is better than the worst fitness value in the population, then the worst member is replaced by the new one.

2.9. Stopping condition

The stopping rule is usually either the number of iterations, maximum number of iterations after last improvement, or a maximum CPU time. Our algorithm terminates when it repeat $n + p$ iterations after last improvement and the best solution has not changed.

Table 1
The results of GApMP algorithm for test problems with positive weights

Test #	n	p	Optimal	GA	Last improvement iteration	%Error		CPU time (s)
						Best	Worst	
1	100	5	5819	5819	61	0.00	46.64	0.35
2	100	10	4093	4093	148	0.00	17.43	0.65
3	100	10	4250	4250	170	0.00	13.49	0.70
4	100	20	3034	3034	214	0.00	9.53	0.85
5	100	33	1355	1355	204	0.00	8.63	0.95
6	200	5	7824	7824	92	0.00	15.74	0.80
7	200	10	5631	5631	156	0.00	7.55	1.50
8	200	20	4445	4445	295	0.00	2.50	3.33
9	200	40	2734	2739	404	0.18	0.80	5.00
10	200	67	1255	1260	311	0.40	2.34	6.33
11	300	5	7696	7696	160	0.00	13.11	6.33
12	300	10	6634	6634	312	0.00	6.61	9.63
13	300	30	4374	4374	552	0.00	1.47	17.33
14	300	60	2968	2969	722	0.03	0.54	30.33
15	300	100	1729	1736	618	0.40	0.64	36.00
16	400	5	8162	8162	175	0.00	16.59	19.66
17	400	10	6999	6999	402	0.00	6.60	32.33
18	400	40	4809	4811	722	0.04	0.76	58.33
19	400	80	2845	2849	1365	0.14	0.37	104.00
20	400	133	1789	1789	1700	0.00	0.45	148.33
21	500	5	9138	9138	150	0.00	25.68	41.33
22	500	10	8579	8579	633	0.00	4.20	70.33
23	500	50	4619	4619	1047	0.00	0.87	141.00
24	500	100	2961	2961	1627	0.00	0.36	253.33
25	500	167	1828	1831	1531	0.16	0.47	391.66
26	600	5	9917	9917	260	0.00	16.26	89.33
27	600	10	8307	8307	746	0.00	3.76	138.66
28	600	60	4498	4498	1758	0.00	0.31	311.66
29	600	120	3033	3034	1720	0.03	0.38	568.00
30	600	200	1989	1995	1932	0.30	0.65	957.66
31	700	5	10,086	10,086	265	0.00	21.13	152.00
32	700	10	9297	9297	760	0.00	7.01	188.00
33	700	70	4700	4703	2113	0.06	0.40	543.66
34	700	140	3013	3013	2081	0.00	0.33	1120.00
35	800	5	10,400	10,400	349	0.00	17.46	216.66
36	800	10	9934	9934	811	0.00	4.45	302.33
37	800	80	5057	5058	2586	0.02	0.31	1111.33
38	900	5	11,060	11,060	436	0.00	14.41	316.33
39	900	10	9423	9423	756	0.00	6.96	487.66
40	900	90	5128	5129	3793	0.02	0.26	2120.00

2.10. The algorithm

The ideas of the previous sections lead to the following algorithm.

Algorithm (*GApMP*)

Initialization:
Generate an initial population P of size $n+p$ as described in Section 2.4.
Find the best member and its fitness value, f_{best} and worst member and its fitness value, f_{worst} in the P.
Let J be the set of all genes.
Iteration counter $r:=0$, $r_{best}:=0$.

Table 2
The average total CPU times for problems (P1) and (P2)

Test #	n	p	(P1)		(P2)	
			GA	MVNS	GA	MVNS
1	100	5	1.000	1.000	1.000	0.400
2	100	10	1.550	1.400	1.500	1.200
3	100	10	1.333	1.400	1.500	1.200
4	100	20	1.777	1.800	1.777	1.800
5	100	33	1.777	2.200	1.555	2.600
6	200	5	2.666	4.000	2.444	3.000
7	200	10	3.000	4.600	2.888	4.600
8	200	20	4.777	10.400	4.111	8.400
9	200	40	7.222	18.000	5.777	17.600
10	200	67	8.222	30.200	6.777	26.000
11	300	5	9.777	12.400	7.777	9.000
12	300	10	11.888	20.600	11.666	18.400
13	300	30	21.777	69.400	18.444	55.800
14	300	60	38.625	118.400	36.125	93.800
15	300	100	35.777	169.400	32.444	138.800
16	400	5	25.333	29.600	24.777	25.000
17	400	10	41.500	47.800	37.000	40.200
18	400	40	63.888	200.200	58.111	174.600
19	400	80	105.777	367.800	86.333	302.600
20	400	133	151.555	545.000	117.888	444.800
21	500	5	49.777	37.200	43.375	29.200
22	500	10	80.777	82.400	75.666	70.000
23	500	50	143.333	432.000	127.111	365.000
24	500	100	242.000	764.800	218.222	621.200
25	500	167	363.222	1189.600	279.444	978.600
26	600	5	102.000	68.200	102.222	60.8
27	600	10	165.666	161.200	141.666	133.400
28	600	60	336.777	825.600	284.888	707.200
29	600	120	540.444	1451.200	531.888	1229.000
30	600	200	817.111	2142.200	734.000	1812.800
31	700	5	162.777	74.000	141.429	67.000
32	700	10	246.333	183.200	201.111	166.200
33	700	70	607.222	1354.800	558.555	1114.600
34	700	140	1083.555	2411.800	1038.444	1948.800
35	800	5	241.888	119.600	217.222	98.200
36	800	10	395.444	237.400	379.777	219.800
37	800	80	1125.666	1944.600	985.111	1624.200
38	900	5	354.222	215.200	309.333	173.000
39	900	10	474.777	299.000	428.333	261.800
40	900	90	2051.555	2873.800	1612.888	2301.600

Iteration step:

While $r - r_{best} \leqslant n + p$ **do** the following:

1. $r := r + 1$,
2. Select two members x, y randomly from P. Let J_x and J_y be the genes of x and y, respectively.
3. Generate a new member z:
   (a) Set $J_z = J_x \cap J_y$ and compute the fitness value for J_z.
   (b) Find the gene $j \in (J_x \cup J_y) \backslash J_z$ so that by adding j to the J_z we have the greatest improvement in the fitness value of J_z.
   (c) If $|J_z| \leqslant p$ go to step 3b
4. Select two genes one from J_z and one from $J \backslash J_z$, which, when swapped, result in the greatest improvement in the fitness value of J_z.

Table 3
The results for test problems with 5 negative weights for problem (P1)

Test #	n	p	Objective function value			%Error	
			Best	GA	MVNS	GA	MVNS
1	100	5	4681	4730	4681	1.05	0.00
2	100	10	2915	2915	2915	0.00	0.00
3	100	10	2529	2529	2529	0.00	0.00
4	100	20	1432	1432	1450	0.00	1.26
5	100	33	61	61	63	0.00	3.17
6	200	5	7061	7061	7061	0.00	0.00
7	200	10	4812	4846	4812	0.71	0.00
8	200	20	3399	3399	3419	0.00	0.59
9	200	40	1919	1919	1926	0.00	0.36
10	200	67	514	514	522	0.00	1.56
11	300	5	7290	7290	7290	0.00	0.00
12	300	10	6201	6201	6201	0.00	0.00
13	300	30	3798	3798	3798	0.00	0.00
14	300	60	2269	2269	2283	0.00	0.62
15	300	100	1153	1153	1157	0.00	0.35
16	400	5	7787	7787	7801	0.00	0.18
17	400	10	6723	6735	6723	0.00	0.18
18	400	40	4219	4219	4219	0.00	0.00
19	400	80	2420	2420	2420	0.00	0.00
20	400	133	1346	1346	1355	0.00	0.67
21	500	5	8888	8888	8888	0.00	0.00
22	500	10	8230	8230	8304	0.00	0.90
23	500	50	4256	4256	4256	0.00	0.00
24	500	100	2538	2538	2538	0.00	0.00
25	500	167	1394	1394	1405	0.00	0.79
26	600	5	9655	9655	9655	0.00	0.00
27	600	10	8038	8038	8040	0.00	0.02
28	600	60	4043	4043	4043	0.00	0.00
29	600	120	2698	2698	2703	0.00	0.19
30	600	200	1604	1604	1614	0.00	0.62
31	700	5	9909	9909	9910	0.00	0.04
32	700	10	8935	8955	8935	0.22	0.00
33	700	70	4411	4411	4417	0.00	0.14
34	700	140	2605	2605	2620	0.00	0.57
35	800	5	10,230	10,230	10,230	0.00	0.00
36	800	10	9735	9735	9767	0.00	0.33
37	800	80	4723	4723	4728	0.00	0.11
38	900	5	10,860	10,860	10,861	0.00	0.01
39	900	10	9070	9070	9070	0.00	0.00
40	900	90	4862	4862	4863	0.00	0.02
Total						1.98	12.68

5. Exchange this two selected genes and compute the fitness value, f_{cur}, of J_z.
6. **If** the generated member is not in the population and $f_{cur} < f_{worst}$ then **do** the following:
   (a) Replace the worst member by the generated member.
   (b) Update the worst member of the population and its fitness value, f_{worst}.
   (c) **If** $f_{cur} < f_{best}$, set $f_{best} = f_{cur}$ and $r_{best} = r$.

endwhile

In the next section we compare the algorithm GApMP with a heuristic method. This method is modified version of the variable neighborhood search (MVNS) algorithm of Hansen and Mladenovic [14] which is modified for coping with positive and negative weights in [12]. The variable neighborhood search algorithm starts

Table 4
The results for test problems with 10 negative weights for problem (P1)

Test #	n	p	Objective function value			%Error	
			Best	GA	MVNS	GA	MVNS
1	100	5	3611	3611	3611	0.00	0.00
2	100	10	1247	1247	1247	0.00	0.00
3	100	10	1029	1029	1029	0.00	0.00
4	100	20	−52	−52	−49	0.00	5.77
5	100	33	−1143	−1143	−1143	0.00	0.00
6	200	5	6374	6374	6374	0.00	0.00
7	200	10	4095	4095	4095	0.00	0.00
8	200	20	2575	2575	2575	0.00	0.00
9	200	40	1088	1088	1091	0.00	0.28
10	200	67	−204	−204	−196	0.00	3.92
11	300	5	6756	6756	6756	0.00	0.00
12	300	10	5610	5610	5610	0.00	0.00
13	300	30	3193	3193	3193	0.00	0.00
14	300	60	1480	1480	1489	0.00	0.61
15	300	100	635	635	638	0.00	0.47
16	400	5	7426	7426	7426	0.00	0.00
17	400	10	6292	6292	6301	0.00	0.14
18	400	40	3693	3693	3694	0.00	0.03
19	400	80	2013	2013	2016	0.00	0.15
20	400	133	910	910	919	0.00	0.98
21	500	5	8582	8630	8582	0.56	0.00
22	500	10	7765	7765	7775	0.00	0.13
23	500	50	3795	3795	3795	0.00	0.00
24	500	100	2151	2151	2161	0.00	0.46
25	500	167	990	990	1002	0.00	1.21
26	600	5	9400	9400	9400	0.00	0.00
27	600	10	7651	7651	7651	0.00	0.00
28	600	60	3576	3576	3582	0.00	0.17
29	600	120	2359	2359	2363	0.00	0.17
30	600	200	1199	1199	1210	0.00	0.92
31	700	5	9519	9688	9519	1.78	0.00
32	700	10	8362	8418	8362	0.67	0.00
33	700	70	4144	4144	4153	0.00	0.17
34	700	140	2219	2219	2233	0.00	0.63
35	800	5	10,039	10,039	10,039	0.00	0.00
36	800	10	9415	9415	9442	0.00	0.27
37	800	80	4384	4384	4395	0.00	0.25
38	900	5	10,696	10,696	10,754	0.00	0.54
39	900	10	8535	8535	8535	0.00	0.00
40	900	90	4695	4595	4606	0.00	0.24
Total						3.01	17.51

with an initial solution X and finds a local optimum solution by interchanges of vertices in X and $V\backslash X$. Then it moves to a new solution which differs in at most p locations from the previous solution and applies again the interchange procedure to this new solution. This is repeated until no further improvement can be found.

3. Computational results

The GApMP was tested on 40 test problems from the ORLIB library, see Beasley [2] which were slightly modified with respect to negative weights. We chose only weights ± 1. We assigned the negative weight -1 either to only the first 5 or first 10 vertices and also to all odd numbered vertices of the test examples.

In the tables we use the results which are reported in [12] for MVNS. In MVNS we allow that the iteration step repeat 100 times.

Table 5
The results for test problems with assigned negative weights to half of the nodes for problem (P1)

Test #	n	p	Objective function value			%Error	
			Best	GA	MVNS	GA	MVNS
1	100	5	−7651	−7651	−7651	0.00	0.00
2	100	10	−9445	−9445	−9445	0.00	0.00
3	100	10	−12398	−12398	−12398	0.00	0.00
4	100	20	−11507	−11507	−11507	0.00	0.00
5	100	33	−10811	−10811	−10811	0.00	0.00
6	200	5	−9971	− 9971	−9971	0.00	0.00
7	200	10	−10403	−10403	−10402	0.00	0.01
8	200	20	−13912	−13912	−13901	0.00	0.08
9	200	40	−13997	−13997	−13988	0.00	0.06
10	200	67	−12437	−12437	−12420	0.00	0.14
11	300	5	−10271	−10271	−10271	0.00	0.00
12	300	10	−14850	−14850	−14846	0.00	0.03
13	300	30	−13557	−13557	−13553	0.00	0.03
14	300	60	−17676	−17676	−17674	0.00	0.01
15	300	100	−14437	−14437	−14433	0.00	0.03
16	400	5	−10792	−10792	−10792	0.00	0.00
17	400	10	−11583	−11583	−11583	0.00	0.00
18	400	40	−16286	−16286	−16282	0.00	0.02
19	400	80	−14200	−14200	−14195	0.00	0.04
20	400	133	−16362	−16362	−16355	0.00	0.04
21	500	5	−11296	−11296	−11296	0.00	0.00
22	500	10	−16588	−16588	−16588	0.00	0.00
23	500	50	−15272	−15272	−15257	0.00	0.10
24	500	100	−17221	−17221	−17207	0.00	0.08
25	500	167	−17924	−17924	−17919	0.00	0.03
26	600	5	−13060	−13060	−13060	0.00	0.00
27	600	10	−16204	−16204	−16179	0.00	0.15
28	600	60	−22970	−22970	−22969	0.00	0.01
29	600	120	−17796	−17796	−17793	0.00	0.02
30	600	200	−21333	−21333	−21324	0.00	0.04
31	700	5	−11466	−11396	−11466	0.61	0.00
32	700	10	−30465	−30465	−30454	0.00	0.04
33	700	70	−16914	−16914	−16908	0.00	0.04
34	700	140	−23803	−23803	−23795	0.00	0.03
35	800	5	−14709	−14709	−14709	0.00	0.00
36	800	10	−21934	−21934	−21934	0.00	0.00
37	800	80	−21038	−21038	−21034	0.00	0.02
38	900	5	−21059	−21059	−21059	0.00	0.00
39	900	10	−38980	−38980	−38980	0.00	0.00
40	900	90	−19350	−19350	−19345	0.00	0.03
Total						0.61	1.08

Both methods are coded in C++ and run on a Pentium IV with 1700 MHz CPU and 256 megabytes of RAM. We ran 5 times each method for all problems and reported the average results.

In Table 1 we present the results of GApMP algorithm for the case positive weights which can be compared with other methods. In this table the column *last improvement iteration* indicates the number of iteration after which no improvement in the solution is attained. The column with the heading *%Error* indicates the relative error; i.e.,

$$\frac{f - f_{\text{opt}}}{|f_{\text{opt}}|} \cdot 100$$

for the best and worst solutions in the population.

Table 6
The results for test problems with five negative weights for problem (P2)

Test #	n	p	Objective function value			%Error	
			Best	GA	MVNS	GA	MVNS
1	100	5	5324	5324	5324	0.00	0.00
2	100	10	3696	3696	3709	0.00	0.35
3	100	10	3592	3592	3592	0.00	0.00
4	100	20	2460	2460	2465	0.00	0.20
5	100	33	994	994	1010	0.00	1.61
6	200	5	7266	7266	7266	0.00	0.00
7	200	10	5224	5224	5238	0.00	0.27
8	200	20	4034	4034	4034	0.00	0.00
9	200	40	2540	2540	2547	0.00	0.28
10	200	67	1077	1077	1081	0.00	0.37
11	300	5	7440	7440	7440	0.00	0.00
12	300	10	6511	6511	6511	0.00	0.00
13	300	30	4187	4187	4190	0.00	0.07
14	300	60	2781	2785	2781	0.14	0.00
15	300	100	1581	1581	1590	0.00	0.57
16	400	5	7870	7870	7870	0.00	0.00
17	400	10	6849	6849	6853	0.06	0.00
18	400	40	4589	4589	4589	0.00	0.00
19	400	80	2741	2741	2749	0.00	0.29
20	400	133	1703	1703	1709	0.00	0.35
21	500	5	8980	8980	8980	0.00	0.00
22	500	10	8403	8403	8471	0.00	0.81
23	500	50	4520	4520	4520	0.00	0.00
24	500	100	2853	2853	2857	0.00	0.14
25	500	167	1735	1735	1748	0.00	0.75
26	600	5	9719	9719	9719	0.00	0.00
27	600	10	8137	8137	8137	0.00	0.00
28	600	60	4385	4385	4385	0.00	0.00
29	600	120	2965	2965	2977	0.00	0.40
30	600	200	1939	1939	1947	0.00	0.62
31	700	5	9915	9968	9915	0.53	0.00
32	700	10	9179	9179	9179	0.00	0.00
33	700	70	4623	4624	4623	0.02	0.00
34	700	140	2913	2913	2922	0.00	0.31
35	800	5	10,286	10,286	10,286	0.00	0.00
36	800	10	9803	9803	9815	0.00	0.12
37	800	80	4967	4967	4972	0.00	0.10
38	900	5	10,914	10,914	10,914	0.00	0.00
39	900	10	9305	9305	9305	0.00	0.00
40	900	90	5075	5075	5102	0.00	0.53
Total						0.75	8.14

Table 2 contains the average total CPU times in GApMP and MVNS for models (P1) and (P2). In the most cases CPU times of GApMP are less than MVNS. Tables 3–8 show the computational results for the cases that 5, 10 and half of all vertices have the negative weight −1. The column *best* in these tables indicates the best solution obtained by the two methods. The column with the heading *%Error* indicates the relative error respect to f_{best}.

Among the tested 240 problems, for 227 problems the best solutions were obtained by the GApMP method and for 111 by MVNS. The total sum of all %Errors for these methods were, 12.05 and 97.06, respectively.

Table 7
The results for test problems with 10 negative weights for problem (P2)

Test #	n	p	Objective function value			%Error	
			Best	GA	MVNS	GA	MVNS
1	100	5	4826	4826	4826	0.00	0.00
2	100	10	2976	2976	2976	0.00	0.00
3	100	10	3341	3341	3341	0.00	0.00
4	100	20	1854	1854	1854	0.00	0.00
5	100	33	533	533	567	0.00	6.75
6	200	5	6787	6787	6787	0.00	0.00
7	200	10	4949	4949	4949	0.00	0.00
8	200	20	3812	3812	3820	0.00	0.21
9	200	40	2391	2391	2393	0.00	0.08
10	200	67	904	904	909	0.00	0.55
11	300	5	7136	7136	7136	0.00	0.00
12	300	10	6395	6395	6395	0.00	0.00
13	300	30	4011	4011	4011	0.00	0.00
14	300	60	2564	2564	2564	0.00	0.00
15	300	100	1462	1462	1471	0.00	0.62
16	400	5	7624	7624	7624	0.00	0.00
17	400	10	6668	6668	6669	0.00	0.01
18	400	40	4437	4437	4437	0.00	0.00
19	400	80	2633	2633	2634	0.00	0.04
20	400	133	1623	1623	1633	0.00	0.62
21	500	5	8800	8800	8800	0.00	0.00
22	500	10	8291	8291	8360	0.00	0.83
23	500	50	4337	4337	4339	0.00	0.05
24	500	100	2748	2779	2748	0.84	0.00
25	500	167	1636	1636	1644	0.00	0.49
26	600	5	9528	9528	9572	0.00	0.46
27	600	10	8004	8004	8004	0.00	0.00
28	600	60	4296	4296	4296	0.00	0.00
29	600	120	2897	2897	2902	0.00	0.17
30	600	200	1850	1850	1858	0.00	0.43
31	700	5	9820	9820	9820	0.00	0.00
32	700	10	9053	9053	9053	0.00	0.00
33	700	70	4572	4572	4583	0.00	0.24
34	700	140	2835	2835	2852	0.00	0.60
35	800	5	10,142	10,142	10,142	0.00	0.00
36	800	10	9637	9637	9637	0.00	0.00
37	800	80	4875	4875	4884	0.00	0.18
38	900	5	10,800	10,800	10,839	0.00	0.36
39	900	10	9201	9201	9201	0.00	0.00
40	900	90	5026	5026	5044	0.00	0.36
Total						0.84	13.05

Table 8
The results for test problems with assigned negative weights to half of the nodes for problem (P2)

Test #	n	p	Objective function value			%Error	
			Best	GA	MVNS	GA	MVNS
1	100	5	−635	−635	−635	0.00	0.00
2	100	10	−1245	−1245	−1245	0.00	0.00
3	100	10	−1131	−1131	−1131	0.00	0.00
4	100	20	−1477	−1477	−1477	0.00	0.00
5	100	33	−1687	−1687	−1687	0.00	0.00
6	200	5	−1163	−1163	−1135	0.00	2.41
7	200	10	−1360	−1360	−1360	0.00	0.00
8	200	20	−1765	−1765	−1764	0.00	0.06
9	200	40	−2212	−2212	−2212	0.00	0.00
10	200	67	−1815	−1815	−1788	0.00	1.49
11	300	5	−797	−797	−797	0.00	0.00
12	300	10	−1290	−1290	−1285	0.00	0.39
13	300	30	−1709	−1709	−1674	0.00	2.05
14	300	60	−2224	−2224	−2190	0.00	1.53
15	300	100	−2152	−2152	−2151	0.00	0.05
16	400	5	−932	−932	−840	0.00	9.87
17	400	10	−1318	−1254	−1318	4.86	0.00
18	400	40	−2096	−2096	−2096	0.00	0.00
19	400	80	−2119	−2119	−2116	0.00	0.14
20	400	133	−2291	−2291	−2277	0.00	0.61
21	500	5	−687	−687	−687	0.00	0.00
22	500	10	−1111	−1111	−1098	0.00	1.17
23	500	50	−1933	−1933	−1915	0.00	0.93
24	500	100	−2216	−2216	−2186	0.00	1.35
25	500	167	−2376	−2376	−2357	0.00	0.80
26	600	5	−820	−820	−808	0.00	1.46
27	600	10	−1053	−1053	−1053	0.00	0.00
28	600	60	−2119	−2119	−2107	0.00	0.57
29	600	120	−2198	−2198	−2192	0.00	0.27
30	600	200	−2320	−2320	−2279	0.00	1.77
31	700	5	−748	−748	−741	0.00	0.94
32	700	10	−1030	−1030	−934	0.00	9.32
33	700	70	−2009	−2009	−1976	0.00	1.65
34	700	140	−2436	−2436	−2423	0.00	0.54
35	800	5	−855	−855	−855	0.00	0.00
36	800	10	−985	−985	−985	0.00	0.00
37	800	80	−2278	−2278	−2262	0.00	0.70
38	900	5	−607	−607	−607	0.00	0.00
39	900	10	−901	−901	−860	0.00	4.55
40	900	90	−2347	−2347	−2346	0.00	0.04
Total						4.86	44.60

4. Related works

Hosage and Goodchild [15] proposed the first GA for the p-median problem. In their algorithm each solution is represented by a binary string with n digits (genes), where each bit corresponds to a facility index. Each gene (1 or 0) indicates whether the corresponding facility is selected as a median or not. Their encoding does not guarantee that exactly p facilities are selected, so they used a penalty function to impose this constraint. Due to this kind of encoding their algorithm performs rather poorly even on small problems.

Dibble and Densham [10] describe another GA with encoding more suitable for the p-median problem. Each individual of the population has exactly p genes, and each genes represents a facility index. They applied their GA to solve a problem with $n = 150$ and $p = 9$ with population size of 1000 and 150 generations.

They compared the results of their algorithm with the results obtained by the interchange heuristic method of Teitz and Bart [23]. Although their GA took longer processing time, both algorithms produced similar solutions.

Moreno-Perez et al. [18] develop a parallelized GA for the p-median problem, where multiple population groups exist and individuals are exchanged in these groups. They do not report computational results.

Bozkaya et al. [3] also describe a GA for the p-median problem. They report the results of an extensive computational experiment with algorithm parameters. Their algorithm produces solutions better than the solutions of an exchange algorithm.

In most recent application of GA to the p-median problem, Alp et al. [1] propose a fast GA with good results. They used the greedy drop procedure to generate new members. In this procedure, first the chromosomes of parents are merged to produce an infeasible solution with m genes where $m > p$. Then the gene whose dropping produces the best fitness function is dropped. This is repeated until number of genes reaches p. In contrast we use the greedy add procedure to generate new members which results better solutions than greedy drop method. We also apply one step of interchange procedure as mutation operator to improve the quality of the generated solutions. Alp et al. [1] do not use the mutation operator.

5. Summary and conclusion

In this paper we proposed a genetic algorithm for solving the p-median problem with positive and negative weights. The results were compared with those obtained by a variable neighborhood search method and showed that for almost all examples our GA results better solutions.

References

[1] O. Alp, E. Erkut, Z. Drezner, An efficient genetic algorithm for the p-median problem, Ann. Oper. Res. 122 (2003) 21–42.
[2] J.E. Beasley, OR-Library: distributing test problems by electronic mail, J. Oper. Res. Soc. 41 (11) (1990) 1069–1072.
[3] Bozkaya, J. Zhang, E. Erkut, A genetic algorithm for the p-median problem, in: Z. Drezner, H. Hamacher (Eds.), Facility Location: Applications and Theory, Springer, Berlin, 2002.
[4] R.E. Burkard, J. Krarup, A linear algorithm for the pos/neg-weighted 1-median problem on a cactus, Computing 60 (1998) 193–215.
[5] R.E. Burkard, E. Çela, H. Dollani, 2-median in trees with pos/neg weights, Discrete Appl. Math. 105 (2000) 51–71.
[6] R.E. Burkard, J. Fathali, H.T. Kakhki, An ant colony algorithm for the pos/neg weighted p-median problem, SFB Report 322, Institute of Mathematics B, Graz University of Technology, 2004.
[7] R.E. Burkard, J. Fathali, A polynomial time method for the pos/neg weighted 3-median problem on a tree, SFB Report 333, Institute of Mathematics B, Graz University of Technology, 2004.
[8] P. Cappanera, A survey on obnoxious facility location problems, Technical Report: TR-99-11, Dipartimento di Informatica, Universita di Pisa, 1999.
[9] E. Carrizosa, F. Plastria, Location of semi-obnoxious facilities, Stud. Locational Anal. 12 (1999) 1–27.
[10] C. Dibble, P.J. Densham, Generating interesting alternatives in GIS and SDSS using genetic algorithms, in: GIS/LIS Symposium, University of Nebraska, Lincoln, 1993.
[11] H.A. Eiselt, G. Laporte, Objectives in location problems, in: Z. Drezner (Ed.), Facility Location. A Survey of Applications and Methods, Springer, New York, 1995 (Chapter 8).
[12] J. Fathali, H.T. Kakhki, Solving the p-median problem with pos/neg weights by variable neighborhood search and some results for special cases, Eur. J. Oper. Res. 170 (2006) 440–462.
[13] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Menlo Park, CA, 1989.
[14] P. Hansen, N. Mladenovic, Variable neighborhood search for the p-median, Location Sci. 5 (1997) 207–226.
[15] M. Hosage, M.F. Goodchild, Discrete space location–allocation solutions from genetic algorithms, Ann. Oper. Res. 6 (1986) 35–46.
[16] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems. Part II: p-medians, SIAM J. Appl. Math. 37 (1979) 539–560.
[17] A.A. Kuehn, M.J. Hamburger, A heuristic program for locating warehouses, Manage. Sci. 9 (1963) 643–666.
[18] J.A. Moreno-Perez, J.M. Moreno-Vega, N. Mladenovic, Tabu search and simulated annealing in p-median problems, in: Talk at the Canadian Operational Research Society Conference, Montral, 1994.
[19] F. Plastria, Optimal location of undesirable facilities: an overview, Belg. J. Oper. Res. Statist. Comput. Sci. 36 (1996) 109–127.
[20] M.G.C. Resende, R. Werneck, A hybrid heuristic for the p-median problem, J. Heuristics 10 (2004) 59–88.
[21] C.R. Reeves, Genetic algorithms, in: C.R. Reeves (Ed.), Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Press, Oxford, 1993, pp. 151–196, Chapter 4.

[22] E. Rolland, D.A. Schilling, J.R. Current, An efficient tabu search heuristic for the p-median problem, Eur. J. Oper. Res. 96 (1996) 329–342.
[23] M.B. Teitz, P. Bart, Heuristic methods for estimating generalized vertex median of a weighted graph, Oper. Res. 16 (1968) 955–961.
[24] R. Whitaker, A fast algorithm for the greedy interchange for large-scale clustering and median location problems, INFOR 21 (1983) 95–108.